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

- Problem fajciarov
- Scoreboard
- Problem hodujucich divochov

Problem fajciarov

Problem fajciarov

- 4 vlakna

Problem fajciarov

- 4 vlakna
  - Agent
  - Traja fajciari

Problem fajciarov

- 4 vlakna
  - Agent
  - Traja fajciari
- Fajciari v nekonecnej slucke

Problem fajciarov

- 4 vlakna
  - Agent
  - Traja fajciari
- Fajciari v nekonecnej slucke
  - Najprv cakaju na material pre usulanie cigarety

Problem fajciarov

- 4 vlakna
  - Agent
  - Traja fajciari
- Fajciari v nekonecnej slucke
  - Najprv cakaju na material pre usulanie cigarety
  - Ked ma fajciar material, usula si cigaretu a uzije ju

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak
  - Papier

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak
  - Papier
  - Zapalky

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak
  - Papier
  - Zapalky
- Kazdy z fajciarov ma k dispozicii

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak
  - Papier
  - Zapalky
- Kazdy z fajciarov ma k dispozicii nekonecne mnozstvo iba jednej z 3 surovin!

Problem fajciarov

- Agent ma nekonecne mnozstvo 3 ingrediencii
  - Tabak
  - Papier
  - Zapalky
- Kazdy z fajciarov ma k dispozicii nekonecne mnozstvo iba jednej z 3 surovin!
- Jeden ma tabak, druhy papier, treti zapalky

Problem fajciarov

- Agent v nekonecnom cykle

Problem fajciarov

- Agent v nekonecnom cykle
  - Nahodne vyberie 2 z 3 surovin

Problem fajciarov

- Agent v nekonecnom cykle
  - Nahodne vyberie 2 z 3 surovin
  - Da ich k dispozicii fajciarom

Problem fajciarov

- Agent v nekonecnom cykle
  - Nahodne vyberie 2 z 3 surovin
  - Da ich k dispozicii fajciarom
- Podla toho, ake suroviny agent dal k dispozicii

Problem fajciarov

- Agent v nekonecnom cykle
  - Nahodne vyberie 2 z 3 surovin
  - Da ich k dispozicii fajciarom
- Podla toho, ake suroviny agent dal k dispozicii
  - Mal by sa chopit aktivity ten pravy fajciar

Problem fajciarov

- Agent v nekonecnom cykle
  - Nahodne vyberie 2 z 3 surovin
  - Da ich k dispozicii fajciarom
- Podla toho, ake suroviny agent dal k dispozicii
  - Mal by sa chopit aktivity ten pravy fajciar
  - Zobrat suroviny a pokracovat (usulat cigaretu a fajcit)

Problem fajciarov

- Priklad

Problem fajciarov

- Priklad
- Agent dal k dispozicii tabak a papier

Problem fajciarov

- Priklad
- Agent dal k dispozicii tabak a papier
- Preto fajciar, ktory ma nekonecny zdroj zapaliek

Problem fajciarov

- Priklad
- Agent dal k dispozicii tabak a papier
- Preto fajciar, ktory ma nekonecny zdroj zapaliek
  - Zoberie tabak a papier

Problem fajciarov

- Priklad
- Agent dal k dispozicii tabak a papier
- Preto fajciar, ktory ma nekonecny zdroj zapaliek
  - Zoberie tabak a papier
  - Usula cigaretu (a fajci)

Problem fajciarov

- Priklad
- Agent dal k dispozicii tabak a papier
- Preto fajciar, ktory ma nekonecny zdroj zapaliek
  - Zoberie tabak a papier
  - Usula cigaretu (a fajci)
  - Da signal agentovi, ze suroviny spracoval

Problem fajciarov

- Aky to ma vztah k realite virtualneho sveta pocitacov?

Problem fajciarov

- Aky to ma vztah k realite virtualneho sveta pocitacov?
- Agent je OS, ktory prerozdeluje zdroje

Problem fajciarov

- Aky to ma vztah k realite virtualneho sveta pocitacov?
- Agent je OS, ktory prerozdeluje zdroje
- Fajciari su aplikacie, ktore zdroje potrebuju

Problem fajciarov

- Agent – OS, fajciari – aplikacie

Problem fajciarov

- Agent – OS, fajciari – aplikacie
- Synchronizacny problem

Problem fajciarov

- Agent – OS, fajciari – aplikacie
- Synchronizacny problem
  - Ak nejake volne zdroje umoznuju, aby nejaka aplikacia pokracovala vo svojej cinnosti

Problem fajciarov

- Agent – OS, fajciari – aplikacie
- Synchronizacny problem
  - Ak nejake volne zdroje umoznuju, aby nejaka aplikacia pokracovala vo svojej cinnosti, tato aplikacia by sa mala “zobudit” a pokracovat vo svojej praci

Problem fajciarov

- Agent – OS, fajciari – aplikacie
- Synchronizacny problem
  - Ak nejake volne zdroje umoznuju, aby nejaka aplikacia pokracovala vo svojej cinnosti, tato aplikacia by sa mala “zobudit” a pokracovat vo svojej praci
  - Opacne garde

Problem fajciarov

- Agent – OS, fajciari – aplikacie
- Synchronizacny problem
  - Ak nejake volne zdroje umoznuju, aby nejaka aplikacia pokracovala vo svojej cinnosti, tato aplikacia by sa mala “zobudit” a pokracovat vo svojej praci
  - Opacne garde: nesmie sa zobudit ziadna aplikacia, ktora (kvoli nedostatku zdrojov, ktore potrebuje) nemoze pokracovat vo svojej praci

Problem fajciarov

- 3 najcastejsie verzie tohto problemu v literature

Problem fajciarov

- 3 najcastejsie verzie tohto problemu v literature

1. Nemozna verzia
2. Zaujimava verzia
3. Trivialna verzia

Problem fajciarov

- Nemozna verzia

Problem fajciarov

- Nemozna verzia
- Nemozno menit kod agenta

Problem fajciarov

- Nemozna verzia
- Nemozno menit kod agenta (asi bezne nemenime kod jadra OS, ci?)

Problem fajciarov

- Nemozna verzia

- Nemozno menit kod agenta
- Nemozno pouzivat if-else alebo pole semaforov

Problem fajciarov

- Nemozna verzia

- Nemozno menit kod agenta
- Nemozno pouzivat if-else alebo pole semaforov

- Nazov vystihuje riesenie...

Problem fajciarov

- Nemozna verzia

- Nemozno menit kod agenta
- Nemozno pouzivat if-else alebo pole semaforov

- Nazov vystihuje riesenie... Ziadne neexistuje

Problem fajciarov

- Zaujimava verzia

Problem fajciarov

- Zaujimava verzia
- Nemozno menit kod agenta

Problem fajciarov

- Zaujimava verzia
- Nemozno menit kod agenta (nechceme menit OS)

Problem fajciarov

- Zaujimava verzia

- Nemozno menit kod agenta (nechceme menit OS)
- Mozeme pouzivat if-else aj semafory

Problem fajciarov

- Trivialna verzia

Problem fajciarov

- Trivialna verzia

- Agent podla toho, co vybral, signalizuje spravnemu fajciarovi, aby pokracoval

Problem fajciarov

- Trivialna verzia

- Agent podla toho, co vybral, signalizuje spravnemu fajciarovi, aby pokracoval
  - Agent musi vediet, na co to-ktore vlakno caka

Problem fajciarov

- Trivialna verzia

- Agent podla toho, co vybral, signalizuje spravnemu fajciarovi, aby pokracoval
  - Agent musi vediet, na co to-ktore vlakno caka
  - Jadro problemu (ingrediencie a cigarety) postrada zmysel

Problem fajciarov

- Trivialna verzia

- Agent podla toho, co vybral, signalizuje spravnemu fajciarovi, aby pokracoval
  - Agent musi vediet, na co to-ktore vlakno caka
  - Jadro problemu (ingrediencie a cigarety) postrada zmysel
  - Problem sa redukuje na trivialnu synchronizaciu

Problem fajciarov

- Trivialna verzia

- Agent podla toho, co vybral, signalizuje spravnemu fajciarovi, aby pokracoval
  - Agent musi vediet, na co to-ktore vlakno caka - NEREALNE
  - Jadro problemu (ingrediencie a cigarety) postrada zmysel
  - Problem sa redukuje na trivialnu synchronizaciu

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien
- Kazde vlakno bude poskytovat ine dve suroviny

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien
- Kazde vlakno bude poskytovat ine dve suroviny
- Jedna sa o 3 konkurentne vlakna

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien
- Kazde vlakno bude poskytovat ine dve suroviny
- Jedna sa o 3 konkurentne vlakna, ale vzdy iba 1 moze byt aktivne!

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien
- Kazde vlakno bude poskytovat ine dve suroviny
- Jedna sa o 3 konkurentne vlakna, ale vzdy iba 1 moze byt aktivne! → pouzijeme Multiplex(1)

Problem fajciarov

- Agenta budeme modelovat pomocou 3 vlakien
- Kazde vlakno bude poskytovat ine dve suroviny
- Jedna sa o 3 konkurentne vlakna, ale vzdy iba 1 moze byt aktivne! → pouzijeme Multiplex(1)

- !!! Pozor na modelovanie s FIFO semaforom !!!

Problem fajciarov

```
Agent_A():
1)  agentSem.wait()
2)  tobacco.signal()
3)  paper.signal()
```

```
Agent_B():
1)  agentSem.wait()
2)  paper.signal()
3)  match.signal()
```

```
Agent_C():
1)  agentSem.wait()
2)  tobacco.signal()
3)  match.signal()
```

```
Init_agent():
1)  agentSem = Semaphore(1)

2)  tobacco = Semaphore(0)
3)  paper = Semaphore(0)
4)  match = Semaphore(0)
```

Problem fajciarov #1

```
Smoker_M():
1)  tobacco.wait()
2)  paper.wait()
3)  agentSem.signal()
```

```
Smoker_T():
1)  paper.wait()
2)  match.wait()
3)  agentSem.signal()
```

```
Smoker_P():
1)  tobacco.wait()
2)  match.wait()
3)  agentSem.signal()
```

Problem fajciarov #1

Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()

Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()

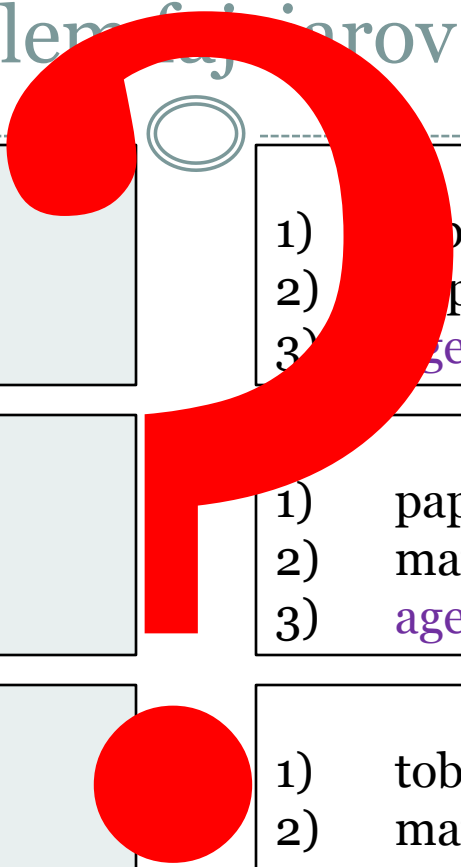
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()

Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()

Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()

Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()

Problem fajčiarov #1



Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()

Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()

Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()

Smoker_M():
1) bacco.wait()
2) per.wait()
3) gentSem.signal()

Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()

Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()

Problem fajciarov #1

Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()

Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()

Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()

Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()

Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()

Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()

Problem fajciarov #1

```
Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()
```

```
Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()
```

```
Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()
```

```
Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()
```

```
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()
```

```
Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()
```

Problem fajciarov #1

Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()

Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()

Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()

Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()

Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()

Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()

Problem fajciarov #1

Agent_A():

1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()

Agent_B():

1) agentSem.wait()
2) paper.signal()
3) match.signal()

Agent_C():

1) agentSem.wait()
2) tobacco.signal()
3) match.signal()

Smoker_M():

1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()

Smoker_T():

1) paper.wait()
2) match.wait()
3) agentSem.signal()

Smoker_P():

1) tobacco.wait()
2) match.wait()
3) agentSem.signal()

Problem fajciarov #1

```
Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()
```

```
Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()
```

```
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()
```

```
Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()
```

```
Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()
```

```
Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()
```

Problem fajciarov #1

```
Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()
```

```
Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()
```

```
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()
```

```
Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()
```

```
Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()
```

```
Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()
```

Problem fajciarov #1

```
Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()
```

```
Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()
```

```
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()
```

```
Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()
```

```
Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()
```

```
Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()
```

Problem fajciarov #1

```
Agent_A():
1) agentSem.wait()
2) tobacco.signal()
3) paper.signal()
```

```
Agent_B():
1) agentSem.wait()
2) paper.signal()
3) match.signal()
```

```
Agent_C():
1) agentSem.wait()
2) tobacco.signal()
3) match.signal()
```

```
Smoker_M():
1) tobacco.wait()
2) paper.wait()
3) agentSem.signal()
```

```
Smoker_T():
1) paper.wait()
2) match.wait()
3) agentSem.signal()
```

```
Smoker_P():
1) tobacco.wait()
2) match.wait()
3) agentSem.signal()
```

Problem fajciarov

- Naivne riesenie nie je spravne

Problem fajciarov

- Naivne riesenie nie je spravne
- Uviaznutie

Problem fajciarov

- Naivne riesenie nie je spravne
- Uviaznutie

- Problem je zaujimavy tym, ze ho nevieme riesit priamo len pomocou agenta a fajciarov

Problem fajciarov

- Naivne riesenie nie je spravne
- Uviaznutie

- Problem je zaujimavy tym, ze ho nevieme riesit priamo len pomocou agenta a fajciarov
- Potrebujeme pomocnych agentov pre fajciarov

Problem fajciarov #2

- Pomocny agent fajciara – diler

Problem fajciarov #2

- Pomocny agent fajciara – diler
- Stara sa o “dodavku” materialu pre konkretneho fajciara

Problem fajciarov #2

- Pomocny agent fajciara – diler
- Stara sa o “dodavku” materialu pre konkretneho fajciara
- Ma na starosti prebudenie fajciara, o ktoreho sa stara

Problem fajciarov #2

- Pomocny agent fajciara – diler
- Stara sa o “dodavku” materialu pre konkretneho fajciara
- Ma na starosti prebudenie fajciara, o ktoreho sa stara
- Sleduje stav “trhu”, a podla toho bud zobudi, alebo nezobudi svojho fajciara

Problem fajciarov #2

- Pomocny agent fajciara – diler v programe

Problem fajciarov #2

- Pomocny agent fajciara – diler v programe
- Reaguje na signal od agenta

Problem fajciarov #2

- Pomocny agent fajciara – diler v programe

- Reaguje na signal od agenta
- Sleduje stav jednotlivych tovarov

Problem fajciarov #2

- Pomocny agent fajciara – diler v programe

- Reaguje na signal od agenta
- Sleduje stav jednotlivych tovarov
- Budi fajciara, ktory moze pokracovat

Problem fajciarov #2

```
Init_agent():
1)    agentSem = Semaphore(1)

2)    tobacco = Semaphore(0)
3)    paper = Semaphore(0)
4)    match = Semaphore(0)
```

Problem fajciarov #2

Smoker_pushers_init():

1) isTobacco = False
2) isMatch = False
3) isPaper = False

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Init_agent():

1) agentSem = Semaphore(1)

2) tobacco = Semaphore(0)
3) paper = Semaphore(0)
4) match = Semaphore(0)

Problem fajciarov #2

```
Smoker_pushers_init():

1)   isTobacco = False
2)   isMatch = False
3)   isPaper = False

4)   tobaccoSem = Semaphore(0)
5)   paperSem = Semaphore(0)
6)   matchSem = Semaphore(0)
```

- premenne typu bool – ci je dany tovar na stole (dostupny)
- tobaccoSem – signalizuje fajciarovi s tabakom, ze moze pokracovat; podobne ostatne semafory

Problem fajciarov #2

```
Smoker_T():

1)  tobaccoSem.wait()
2)  makeCigarette()
3)  agentSem.signal()
4)  smoke()
```

- premenne typu bool – ci je dany tovar na stole (dostupny)
- tobaccoSem – signalizuje fajciarovi s tabakom, ze moze pokracovat; podobne ostatne semafory

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.

Problem fajciarov #2

Pusher_A():

1. tobacco.wait()
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.
5.
6.
7.
8.
9.
10.
11.
```

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.              isPaper ← False
5.
6.
7.
8.
9.
10.
11.
```

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.              isPaper ← False
5.              matchSem.signal()
6.
7.
8.
9.
10.
11.
```

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.              isPaper ← False
5.              matchSem.signal()
6.          elif isMatch:
7.
8.
9.
10.
11.
```

Problem fajciarov #2

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.              isPaper ← False
5.              matchSem.signal()
6.          elif isMatch:
7.              isMatch ← False
8.
9.
10.
11.
```

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.      tobacco.wait()
2.
3.          if isPaper:
4.              isPaper ← False
5.              matchSem.signal()
6.          elif isMatch:
7.              isMatch ← False
8.              paperSem.signal()
9.
10.
11.
```

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.    tobacco.wait()
2.
3.      if isPaper:
4.          isPaper ← False
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch ← False
8.          paperSem.signal()
9.      else:
10.         isTobacco ← True
11.
```

Problem fajciarov #2

Smoker_M():

1) matchSem.wait()
2) makeCigarette()
3) agentSem.signal()
4) smoke()

Pusher_A():

```
1.  tobacco.wait()
2.  mutex.wait()
3.      if isPaper:
4.          isPaper ← False
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch ← False
8.          paperSem.signal()
9.      else:
10.         isTobacco ← True
11. mutex.signal()
```

Problem fajciarov

- Agent musi stale cakat
- Nemoze pokracovat, pokym nezmizne dodany tovar zo stola

```
Agent_XYZ():
1)    agentSem.wait()
2)    ingredient1.signal()
3)    ingredient2.signal()
```

- Vieme vylepsit riesenie tak, aby agent necakal?

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = False
2) isMatch = False
3) isPaper = False

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Pusher_A():

```
1.  tobacco.wait()
2.  mutex.wait()
3.      if isPaper:
4.          isPaper ← False
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch ← False
8.          paperSem.signal()
9.      else:
10.         isTobacco ← True
11. mutex.signal()
```

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Pusher_A():

```
1.  tobacco.wait()
2.  mutex.wait()
3.      if isPaper:
4.          isPaper ← False
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch ← False
8.          paperSem.signal()
9.      else:
10.         isTobacco ← True
11. mutex.signal()
```

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Pusher_A():

```
1.  tobacco.wait()
2.  mutex.wait()
3.      if isPaper:
4.          isPaper -= 1
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch -= 1
8.          paperSem.signal()
9.      else:
10.         isTobacco += 1
11. mutex.signal()
```

Problem fajciarov #3

Smoker_pushers_init():

1) isTobacco = 0
2) isMatch = 0
3) isPaper = 0

4) tobaccoSem = Semaphore(0)
5) paperSem = Semaphore(0)
6) matchSem = Semaphore(0)

Pusher_A():

```
1.  tobacco.wait()
2.  mutex.wait()
3.      if isPaper:
4.          isPaper -= 1
5.          matchSem.signal()
6.      elif isMatch:
7.          isMatch -= 1
8.          paperSem.signal()
9.      else:
10.         isTobacco += 1
11. mutex.signal()
```

Problem fajciarov - vizualizacia

- Majme miestnost

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin
- Diler

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin
- Diler (1 diler nosi do miestnosti iba 1 surovinu!)

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin
- Diler (1 diler nosi do miestnosti iba 1 surovinu!)
  - Po JEDNOM chodia do miestnosti

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin
- Diler (1 diler nosi do miestnosti iba 1 surovinu!)
  - Po JEDNOM chodia do miestnosti
  - Skontroluje stol, ci moze skompletizovat dodavku pre fajc

Problem fajciarov - vizualizacia

- Majme miestnost:
  - Stol
  - Na stolickach spia fajciari ;)
- Agent poveruje dilerov dodavanim surovin
- Diler (1 diler nosi do miestnosti iba 1 surovinu!)
  - Po JEDNOM chodia do miestnosti
  - Skontroluje stol, ci moze skompletizovat dodavku pre fajc
  - Ak ano, zobudi ho; ak nie, necha surovinu na stole a odide

Scoreboard

Scoreboard

- Majme miestnost

Scoreboard

- Majme miestnost:
  - Stol so surovinami

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi
- Agent

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi
- Agent
  - Vojde do mietnosti (po jednom)

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi
- Agent
  - Vojde do mietnosti (po jednom)
  - Skontroluje stav surovin na stole

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi
- Agent
  - Vojde do mietnosti (po jednom)
  - Skontroluje stav surovin na stole
  - Na zaklade stavu vyvola aktivitu

Scoreboard

- Majme miestnost:
  - Stol so surovinami
  - Spiacich ludi
- Agent
  - Vojde do mietnosti (po jednom)
  - Skontroluje stav surovin na stole
  - Na zaklade stavu vyvola aktivitu (napr. zobudi nejakeho spaca)

Scoreboard

- Scoreboard – vzor, ktory budeme aj nadalej vyuzivat

Scoreboard

- Scoreboard – vzor, ktory budeme aj nadalej vyuzivat

- Ciselne premenne tvoria stav systemu

Scoreboard

- Scoreboard – vzor, ktory budeme aj nadalej vyuzivat

- Ciselne premenne tvoria stav systemu
- Vlakna po jednom (cez mutex) kontroluju stav systemu

Scoreboard

- Scoreboard – vzor, ktory budeme aj nadalej vyuzivat

- Ciselne premenne tvoria stav systemu
- Vlakna po jednom (cez mutex) kontroluju stav systemu
- Na zaklade stavu vyvolavaju zelanu aktivitu

Problem hodujucich divochov

Problem hodujucich divochov

- Kmen divochov ma klasicke hody

Problem hodujucich divochov

- Kmen divochov ma klasicke hody
- Jeden spolocny hrniec, z ktoreho si kazdy sam berie (ak hrniec nie je prazdny!)

Problem hodujucich divochov

- Kmen divochov ma klasicke hody
- Jeden spolocny hrniec, z ktoreho si kazdy sam berie (ak hrniec nie je prazdny!)
- Hrniec dokaze poňať M porcii duseneho misionara

Problem hodujucich divochov

- Kmen divochov ma klasicke hody
- Jeden spolocny hrniec, z ktoreho si kazdy sam berie (ak hrniec nie je prazdny!)
- Hrniec dokaze ponat M porcii duseneho misionara
- Ak je hrniec prazdny

Problem hodujucich divochov

- Kmen divochov ma klasicke hody
- Jeden spolocny hrniec, z ktoreho si kazdy sam berie (ak hrniec nie je prazdny!)
- Hrniec dokaze ponat M porcii duseneho misionara
- Ak je hrniec prazdny:
  - Divoch, ktory si prave chcel nabrat, zobudi kuchara,
  - A caka, kym kuchar nenaplni hrniec

Problem hodujucich divochov

Nesynchronizovany kod
savage():

```
1)    while True:
2)        getServingFromPot()
3)        eat()
```

Nesynchronizovany kod
cook():

```
1)    while True:
2)        putServingsInPot()
```

- Synchronizacne obmedzenia
  - Divoch nesmie zobrat porciu, ak je hrniec prazdny
  - Kuchar smie naplnit hrniec, iba ak je tento prazdny

Problem hodujucich divochov

- Mozeme pouzit pristup ako v pripade synchronizacneho problemu konzument-producent

Problem hodujucich divochov

- Mozeme pouzit pristup ako v pripade synchronizacneho problemu konzument-producent
- Pouzime scoreboard!
  - Pocet porcii
  - Zobudenie kuchara
  - Cakanie na signal doplnenia hrnca

init():

1) servings ← 0
2) mutex ← Mutex()
3) emptyPot ← Event()
4) fullPot ← Event()

emptyPot – hrniec je prazdny

fullPot – hrniec je plny

Problem hodujucich divochov

savage():

1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

cook():

1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

Problem hodujucich divochov

savage():

1) while True:
2)
3)
4)
5)
6)
7)
8)
9)
10)

cook():

1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

Problem hodujucich divochov

savage():

```
1)      while True:
2)
3)
4)
5)
6)
7)
8)
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)
3)
4)
5)
6)
7)          getServingFromPot()
8)
9)
10)       eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)
4)
5)
6)
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)
4)
5)
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)
5)
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)
3)
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)
3)              putServingsInPot()
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
6)
7)
8)
9)
10)
```

Problem hodujucich divochov

savage():

```
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)
10)         eat()
```

cook():

```
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()
  2. Moze nastat uviaznutie?

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()
  2. Moze nastat uviaznutie? V akom pripade?

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()
  2. Moze nastat uviaznutie? V akom pripade?
  3. Aky je pristup do hrnca vzhladom na vlakna?

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()
  2. Moze nastat uviaznutie? V akom pripade?
  3. Aky je pristup do hrnca vzhladom na vlakna? Su funkcie getServing... a putServings... bezpecne?

Problem hodujucich divochov

```
savage():
1)      while True:
2)          mutex.lock()
3)              if servings == 0:
4)                  emptyPot.signal()
5)                  fullPot.wait()
6)              servings -= 1
7)              getServingFromPot()
8)          mutex.unlock()
9)          eat()
```

```
cook():
1)      while True:
2)              emptyPot.wait()
3)              putServingsInPot()
4)              servings ← M
5)              fullPot.signal()
```

- Rozne otazky...
  1. R4 cook()
  2. Moze nastat uviaznutie? V akom pripade?
  3. Aky je pristup do hrnca vzhladom na vlakna? Su funkcie getServing... a putServings... bezpecne?
  4. Musi byt getServingFromPot v KO?