Семафори задачи + решения :

Зад 1. 27-8-2016-СЕ

Задача: Всеки от процесите `P`, `Q` и `R` изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на `P`, `Q` и `R` така, че да се изпълнят следните изисквания:

(а) Инструкция `p_1` да се изпълни преди `q_2` и `r_2`.
(б) Ако `q_2` се изпълни преди `r_2`, то и `q_3` да се изпълни преди `r_2`.
(в) Ако `r_2` се изпълни преди `q_2`, то и `r_3` да се изпълни преди `q_2`.

Решение

За синхронизация използваме семафор `t`, инициализираме го с блокиращо начално състяние:

```
semaphore t
t.init(0)
```

Добавяме в кода на процесите `P`, `Q` и `R` синхронизиращи инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  t.signal()         t.wait()           t.wait()
  p_2                q_2                r_2
  p_3                q_3                r_3
                     t.signal()         t.signal()
```

Зад 2. 13-05-2017-КН

Задача: Преди стартиране на процесите `P` и `Q` са инициализирани два семафора и брояч:

```
semaphore e, m
e.init(1); m.init(1)
int cnt = 0
```

Паралелно работещи копия на `P` и `Q` изпълняват поредица от инструкции:

```
process P                  process Q
  m.wait()                   e.wait()
   cnt=cnt+1                  q_section
   if cnt=1 e.wait()         e.signal()
  m.signal()
  p_section
  m.wait()
   cnt=cnt-1
   if cnt=0 e.signal()
  m.signal()
```

Дайте обоснован отговор на следните въпроси:

(а) Могат ли едновременно да се изпълняват инструкциите `p_section` и `q_section`?

(б) Могат ли едновременно да се изпълняват няколко инструкции `p_section`?

(в) Могат ли едновременно да се изпълняват няколко инструкции `q_section`?

(г) Има ли условия за deadlock или starvation за някой от процесите?

Упътване:

Ще казваме, че `P` е в критична секция, когато изпълнява инструкцията си `p_section`. Същото за `Q`, когато изпълнява `q_section`.

Изяснете смисъла на брояча `cnt` и какви процеси могат да бъдат приспани в опашките на двата семафора.

Покажете, че в опашката на семафора `e` има най-много едно копие на `P` и произволен брой копия на `Q`.

Покажете, че в момента на изпълнение на `e.signal()` в кой да е от процесите, никой процес не е в критичната си секция.

Решение

Първо забелязаме, че семафорът `m` се ползва само от `P` в ролята на mutex. В неговата опашка може да има само копия на `P` и само едно работещо копие може да намалява/увеличава брояча синхронизирано с блокирането/освобождаването на семафора `e`.

Увеличаването на `cnt` става преди критичната секция на `P`, а намаляването след нея. Ако не вървят никакви копия на `Q`, лесно се убеждаваме, че могат да се изпълняват произволен брой критични секции на `P`, като броячът съвпада с броя на паралелно изпълняваните критични секции. Така отговорът на въпрос (б) е ДА.

Заемането на семафора `e` в `P` става точно когато `cnt` променя стойността си от 0 в 1. Освобождаването става точно когато `cnt` променя стойността си от 1 в 0.

Тъй като при инициализацията броячът на `e` е 1, а употребата му и в двата вида процеси започва със заемане и завършва с освобождаване, само едно копие от двата типа ще може да премине `e.wait()`. Разглеждаме два случая:

(А) Процесът `Q` преминава. Тогава ще се изпълни критичната му секция, но само от това копие. Останалите копия на `Q` ще бъдат приспани от първата си инструкция. Следователно отговорът на въпрос (в) е НЕ.

Ако версия на `P` пробва `e.wait()`, тя също ще бъде приспана. Това ще стане точно когато `cnt` променя стойността си от 0 в 1, тоест не се изпълняват критични секции на `P`. В момента на приспиване и мутекса `m` е блокиран. Това обстоятелство ще блокира всички опити на други копия на `P` да преминат `m`. В този случай в опашката на семафора `e` има точно едно копие на `P`.

(В) Процесът `P` преминава. Ще започне изпълнение на неговата критична секция и евентуално на други копия на `P`, докато `cnt>0`. През този период всички копия на `Q` ще бъдат приспани от първата си инструкция. Когато `cnt` намалее до 0, никое копие не изпълнява критична секция.

От двата разгледани случая следва, че в един момент могат да се изпълняват няколко критични секции на `P` или една критична секция на `Q`. Следователно отговорът на въпрос (а) е НЕ.

В описаната схема няма условия за deadlock. `Q` не може да инициира deadlock, тъй като ползва само един ресурс. `P` също не може поради реда на заемане на ресурсите (първо заема семафора `m`, после `e`).

В описаната схема има условия за гладуване (starvation) на процес `Q`. Нека критичната секция на `P` се изпълнява бавно и `Q` започва работа след `P`. Ще започне изпълнение на критична секция на `P` и ако постоянно започват работа нови копия, броячът `cnt` може да остане положителен неограничено време. Така `Q` ще бъде приспан неограничено дълго.

Зад 3. 26-08-2017

Задача 5. Всеки от процесите P и Q изпълнява поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на P и Q така, че инструкция p_1 да се изпълни преди q_2, а q_1 да се изпълни преди p_2.

Решение:

Задача 5. За двете искани в условието синхронизации използваме два семафора – t1 и t2, инициализираме ги с блокиращо начално състояние:

```
semaphore t1,t2
t1.init(0)
t2.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  p_1                      q_1
  t1.signal()              t2.signal()
  t2.wait()                t1.wait()
  p_2                      q_2
```

Зад 4. 26-08-2017

Задача 6. Паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на работещите копия така че:

а) Във произволен момент от времето да работи най-много едно от копията.
б) Работещите копия да се редуват във времето – след изпълнение на копие на P, да следва изпълнение на копие на Q, и обратно.
в) Първоначално е разрешено да се изпълни копие на P.

Решение:

Задача 6. Използваме два семафора – s_p и s_q, инициализираме ги така:

```
semaphore s_p, s_q
s_p.init(1)
s_q.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  s_p.wait()               s_q.wait()
  p_1                      q_1
  p_2                      q_2
  s_q.signal()             s_p.signal()
```

Зад 5. 2018-КН

Задача за КН1: Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P                process Q                process R
  p_1                      q_1                      r_1
  p_2                      q_2                      r_2
  p_3                      q_3                      r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят следните изисквания:

(а) Инструкция p_1 да се изпълни преди q_2 и r_2.

(б) Инструкция r_2 да се изпълни преди p_3.

Забележка: Решение с 2 семафора ще бъде оценено с 30 точки, решение с повече семафори ще ви донесе 20 точки.

Решение:

```
semaphore s1,s2;
s1.init(0)
s2.init(0)

process P                process Q                process R
   p1                       q1                       r1
   s1.signal()              s1.wait()                s1.wait()
   p2                       q2                       r2
   s2.wait()                s1.signal()              s2.signal()
   p3                       q3                       s1.signal()
                                                     r3
```

Зад 6. 2018-КН2

Задача за КН2: Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P                process Q                process R
  p_1                      q_1                      r_1
  p_2                      q_2                      r_2
  p_3                      q_3                      r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят следните изисквания:

(а) Инструкция p_1 да се изпълни преди q_2.

(б) Инструкция q_1 да се изпълни преди r_2.

(в) Инструкция r_1 да се изпълни преди p_2.

(г) Инструкция r_3 да се изпълни след p_2 и q_2.

Зад 7. 2018-СИ

Задача за СИ: Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
  p_3                q_3                r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят едновременно следните изисквания:

(а) Някоя от инструкциите p_2 и q_2 да се изпълни преди r_2.

(б) Ако инструкция p_2 се изпълни преди r_2, то q_2 да се изпълни след r_2.

(в) Ако инструкция q_2 се изпълни преди r_2, то p_2 да се изпълни след r_2.

Решение:

За синхронизация използваме семафори f и u, инициализираме ги така:

```
semaphore f, u
f.init(1)
u.init(0)
```

Добавяме в кода на процесите P, Q и R синхронизиращи инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  f.wait()           f.wait()           u.wait()
  p_2                q_2                r_2
  u.signal()         u.signal()         f.signal()
  p_3                q_3                r_3
```

Зад 8. 24-08-2018-КН

Задача 6: Всеки от процесите P и Q изпълнява поредица от три инструкции:

```
process P          process Q
  p_1                q_1
  p_2                q_2
  p_3                q_3
```

Осигурете чрез два семафора синхронизация на P и Q така, че отделните инструкции да се изпълнят в следния времеви ред:

p_1, q_1, p_2, q_2, p_3, q_3

Задача 6:

Използваме семафорите t1 и t2, инициализираме ги така:

```
semaphore t1, t2
t1.init(1)
t2.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  t1.wait()                t2.wait()
  p_1                      q_1
  t2.signal()              t1.signal()
  t1.wait()                t2.wait()
  p_2                      q_2
  t2.signal()              t1.signal()
  t1.wait()                t2.wait()
  p_3                      q_3
  t2.signal()              t1.signal()
```

Зад 9. 25-08-2018-СИ

Задача 1, СИ. Множество паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от три инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
  p_3                      q_3
```

Осигурете чрез семафори синхронизация на работещите копия, така че:

Три инструкции – p_1, q_2 и p_3 се редуват циклично.

Първа се изпълнява инструкция p_1 на някое от работещите копия на процес P. След завършването ѝ се изпълнява инструкция q_2 на някое копие на Q, а след нея – p_3 на копие на P. С това цикълът завършва и отново може да се изпълни инструкция p_1 на някое от работещите копия на процес P.

Решение:

Задача 1, СИ За исканите в условието синхронизации използваме три семафора – t1, t2 и t3, инициализираме ги така:

```
semaphore t1, t2, t3
t1.init(1)
t2.init(0)
t3.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  t1.wait()                q_1
  p_1                      t2.wait()
  t2.signal()              q_2
  p_2                      t3.signal()
  t3.wait()                q_3
  p_3
  t1.signal()
```

Зад 10. 23-03-2019-КН

Задача 4. КН1 Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P                process Q                process R
  p_1                      q_1                      r_1
  p_2                      q_2                      r_2
  p_3                      q_3                      r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да са изпълнение едновременно условията:
(1) инструкция p_1 да се изпълни преди q_2 и r_2.
(2) инструкция p_3 да се изпълни след q_2 и r_2.

Решение:

Задача 4. КН1 За синхронизация използваме семафори s, t и u, инициализираме ги така:

```
semaphore s, t, u
s.init(0)
t.init(0)
u.init(0)
```

Добавяме в кода на процесите P, Q и R синхронизиращи инструкции:

```
process P                process Q                process R
  p_1                      q_1                      r_1
  s.signal()               s.wait()                 s.wait()
  p_2                      s.signal()               s.signal()
  t.wait()                 q_2                      r_2
  u.wait()                 t.signal()               u.signal()
  p_3                      q_3                      r_3
```

Зад 11. 23-03-2019-КН

Задача 4. КН2 Всеки от процесите P, Q и R изпълнява поредица от две инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
```

Осигурете чрез три семафора синхронизация на P, Q и R така, че отделните инструкции да се изпълнят в следния времеви ред:
p_1, q_1, r_1, p_2, q_2, r_2

Решение:

Задача 4. КН2 Използваме семафорите t1, t2 и t3, инициализираме ги така:

```
semaphore t1, t2, t3
t1.init(1)
t2.init(0)
t3.init(0)
```

Добавяме в кода на процесите синхронизиращи инструкции:

```
process P                process Q                process R
  t1.wait()                t2.wait()                t3.wait()
  p_1                      q_1                      r_1
  t2.signal()              t3.signal()              t1.signal()
  t1.wait()                t2.wait()                t3.wait()
  p_2                      q_2                      r_2
  t2.signal()              t3.signal()              t1.signal()
```

Зад 12. 23-03-2019-СИ

Задача 4. СИ Всеки от процесите P и Q изпълнява поредица от три инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
  p_3                      q_3
```

Осигурете чрез два семафора синхронизация на P и Q така, че да са изпълнени едновременно следните времеви зависимости:
(1) инструкция p_1 да се изпълни преди q_2
(2) инструкция q_2 да се изпълни преди p_3
(3) инструкция q_1 да се изпълни преди p_2
(4) инструкция p_2 да се изпълни преди q_3

Решение:

```
semaphore t1,t2
t1.init(0)
t2.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  p_1                      q_1
  t1.signal()              t2.signal()
  t2.wait()                t1.wait()
  p_2                      q_2
  t1.signal()              t2.signal()
  t2.wait()                t1.wait()
  p_3                      q_3
```

Зад 13. 23-03-2019-КН

Задача 4. КН2 Всеки от процесите P, Q и R изпълнява поредица от две инструкции:

```
process P         process Q         process R
  p_1               q_1               r_1
  p_2               q_2               r_2
```

Осигурете чрез три семафора синхронизация на P, Q и R така, че отделните инструкции да се изпълнят в следния времеви ред:
p_1, q_1, r_1, p_2, q_2, r_2

Решение:

Задача 4. КН2 Използваме семафорите t1, t2 и t3, инициализираме ги така:

```
semaphore t1, t2, t3
t1.init(1)
t2.init(0)
t3.init(0)
```

Добавяме в кода на процесите синхронизиращи инструкции:

```
process P                 process Q                 process R
  t1.wait()                 t2.wait()                 t3.wait()
  p_1                       q_1                       r_1
  t2.signal()               t3.signal()               t1.signal()
  t1.wait()                 t2.wait()                 t3.wait()
  p_2                       q_2                       r_2
  t2.signal()               t3.signal()               t1.signal()
```

Зад 14. 23-03-2019-КН2

Задача 4. КН1 Всеки от процесите P, Q и R изпълнява поредица от три инструкции:

```
process P                 process Q                 process R
  p_1                       q_1                       r_1
  p_2                       q_2                       r_2
  p_3                       q_3                       r_3
```

Осигурете чрез семафори синхронизация на P, Q и R така, че да са изпълнение едновременно условията:
(1) инструкция p_1 да се изпълни преди q_2 и r_2.
(2) инструкция p_3 да се изпълни след q_2 и r_2.

Решение:

```
semaphore s, t, u
s.init(0)
t.init(0)
u.init(0)
```

Добавяме в кода на процесите P, Q и R синхронизиращи инструкции:

```
process P                process Q                process R
  p_1                      q_1                      r_1
  s.signal()               s.wait()                 s.wait()
  p_2                      s.signal()               s.signal()
  t.wait()                 q_2                      r_2
  u.wait()                 t.signal()               u.signal()
  p_3                      q_3                      r_3
```

Зад 15. 25-08-2019-СИ

Задача 1, СИ, 25.08

Множество паралелно работещи копия на процеса P изпълняват поредица от две инструкции:

```
process P
  p_1
  p_2
```

Осигурете чрез семафори синхронизация на работещите копия, така че:

Инструкцията p_2 на всяко от работещите копия да се изпълни след като инструкция p_1 е завършила изпълнението си в поне 3 работещи копия.

Упътване: Освен семафори, ползвайте и брояч.

Решение:

За исканите в условието синхронизации използваме брояч cnt и два семафора – m1 и m2, инициализираме ги така:

```
semaphore m1, m2
m1.init(1)
m2.init(0)
int cnt=0
```

Добавяме в кода на процеса P синхронизиращи инструкции:

```
process P
  p_1
  m1.wait()
   cnt=cnt+1
   if cnt=3 m2.signal()
  m1.signal()
  m2.wait()
  m2.signal()
  p_2
```

Зад 16. 28-06-2020-СИ

Задача 1. Множество паралелно работещи копия на всеки от процесите P и Q изпълняват поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на P и Q, така че поне една инструкция p_1 да се изпълни преди всички q_2, и поне една инструкция q_1 да се изпълни преди всички p_2.

Решение:

Задача 1. За двете искани в условието синхронизации използваме два семафора – t1 и t2, инициализираме ги с блокиращо начално състяние:

```
semaphore t1,t2
t1.init(0)
t2.init(0)
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  p_1                      q_1
  t1.signal()              t2.signal()
  t2.wait()                t1.wait()
  t2.signal()              t1.signal()
  p_2                      q_2
```

Или “прецизно решение”

```
semaphore t1,t2,m
t1.init(0)
t2.init(0)
m.init(1)
int c1=0, c2=0
```

Добавяме в кода на процесите P и Q синхронизиращи инструкции:

```
process P                process Q
  p_1                      q_1
  m.wait()                 m.wait()
  if c1=0                  if c2=0
     c1=1                     c2=1
     t1.signal()              t2.signal()
  m.signal()               m.signal()
  t2.wait()                t1.wait()
  t2.signal()              t1.signal()
  p_2                      q_2
```

Зад 17. 19-06-2021

Задача 1, (20 точки)

Всеки от процесите P, Q и R изпълнява поредица от инструкции:

```
process P          process Q          process R
  p_1                q_1                r_1
  p_2                q_2                r_2
```

Процесите P и Q са единични, процесът R се изпълнява в много копия.

Осигурете чрез семафори синхронизация на P, Q и R така, че да се изпълнят едновременно следните изисквания:

(а) Всички инструкции на P и Q да се изпълнят преди инструкция r_1 на всяко копие на R.

(б) Процесите P и Q да се изпълнят ефикасно, т.е. да е възможно паралелното им изпълнение, без да се изчакват.

Решение:

Задача 1. решение 2 За синхронизация използваме семафори sp и sq, инициализираме ги така:

```
semaphore sp, sq
sp.init(0)
sq.init(0)
```

Добавяме в кода на процесите P, Q и R синхронизиращи инструкции:

```
process P          process Q          process R
  p_1                q_1                sp.wait()
  p_2                q_2                sp.signal()
  sp.signal()        sq.signal()        sq.wait()
                                        sq.signal()
                                        r_1
                                        r_2
```

Задача 1. решение 3 За синхронизация използваме семафори sp и sq, инициализираме ги така:

```
semaphore sp, sq
sp.init(0)
sq.init(0)
```

Добавяме в кода на процесите P, Q и R синхронизиращи инструкции:

```
process P          process Q          process R
  p_1                q_1                sq.wait()
  p_2                q_2                sq.signal()
  sp.signal()        sp.wait()          r_1
                     sq.signal()        r_2
```