Теоретична задача за курса 'Операционни системи', СУ, ФМИ, 13.05.2017 г.

Име: _____ ФН: _____ Спец.: __ Курс: __ Група: __

Забележка: Задачата дава 40 точки към общия сбор точки!

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.