Държавен изпит за дисциплина “Операционни системи”, СУ, ФМИ, 10.09.2018 г.

**Задача 1.** 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`.

**Задача 2.** 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 работещите копия, така че:

а) В 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`.

*Упътвания:*

(1) Семафорът е обект за синхронизация, локалните му данни са брояч $cnt$ и списък на приспаните процеси $L$. Конструкторът му $init(n)$ присвоява начална стойност на брояча ($cnt = n$), списъкът се инициалира празен.

Семафорът има два метода – $wait()$ и $signal()$.

Методът $wait()$ намаля с единица брояча $cnt$ и ако стойността на брояча стане отрицателна, добавя в списъка $L$ информация за текущия процес и го спира временно (процесът бива приспан, блокиран).

Методът $signal()$ увеличава $cnt$ и ако стойността на брояча преди увеличението е отрицателна, изважда процес от списъка $L$ и го събужда. Ако от $L$ се вади най-рано приспания процес, наричаме семафора силен. Всяка друга стратегия на събуждане реализира слаб семафор.

(2) Приемете, че инициализацията на семафорите се прави от процес, който поражда процесите, обсъждани в условието на задачата, преди тяхното стартиране.

## Примерни решения

**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()
  p_2                      q_2
```

Инструкцията `q_2` ще се изпълни, след като процесът `Q` премине бариерата `t1.wait()`. Това се случва след изпълнението от `P` на ред `t1.signal()`, който следва инструкция `p_1`.

Аналогично, инструкцията `p_2` ще се изпълни след изпълнението на ред `t2.signal()`, който следва инструкция `q_1`.

Решението на задачата осигурява среща във времето (rendezvous) на двата процеса. Важен е редът на извикване на инструкциите, управляващи семафорите. Ако го обърнем, получаваме класически пример за deadlock:

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

**Задача 2.** 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()
```