Име: _____ ФН: _____ Група: ____

Теоретични задачи по ОС за специалност КН, 16.08.2022 г.:

Задача 1, (15 точки)

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:

1. В произволен момент от времето да работи най-много едно от копията.
2. Работещите копия да се редуват във времето – след изпълнение на копие на `P`, два пъти се изпълнява копие на `Q`, и цикълът се повтаря – веднъж `P`, два пъти `Q` и т.н.
3. Първоначално е разрешено да се изпълни копие на `P`.

Упътване: Освен семафори, ползвайте и брояч.

Задача 2, (15 точки)

Опишете накратко основните процедури и структури данни, необходими за реализация на семафор.

Каква е разликата между слаб и силен семафор?

Опишете максимално несправедлива ситуация, която може да се получи в избирателна секция, ако на входа на секцията пазач – член на изборната комисия пуска гласоподавателите вътре така:

(1) във всеки момент в секцията може да има най-много двама гласоподаватели.

(2) пазачът работи като слаб семафор.

Примерни решения

Задача 1. Използваме два семафора – `s_p` и `s_q`, и брояч `cnt`, инициализираме ги така:

```
semaphore s_p, s_q
s_p.init(1)
s_q.init(0)
int cnt=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()               cnt=(cnt+1) mod 2
                             if (cnt=0)
                                then s_p.signal()
                                else s_q.signal()
```

Задача 2. *Структурите* данни, необходими за реализация на семафор са:

Брояч `cnt`, в който се пази броя на процесите, които могат да бъдат допуснати до ресурса, охраняван от семафора.

Контейнер `Q`, в който се пази информация кои процеси чакат достъп до ресурса.

Процедурите, необходими за реализация на семафор са:

Конструктор `Init(c0:integer)`, който задава начална стойност на брояча `cnt`. Контейнерът `Q` се инициализира да е празен.

Метод `Wait()`, който се ползва при опит за достъп до ресурса (заемане на ресурса). Броячът се намалява с единица и ако стане отрицателен, процесът викащ `Wait()` се блокира, а номерът му се вкарва в контейнера `Q`.

Метод `Signal()`, който се ползва при завършване на достъпа до ресурса (освобождаване на ресурса). Броячът се увеличава с единица и ако `Q` не е празен, един от процесите в него се вади и активира.

Семафорът е *силен*, когато контейнера `Q` е реализиран като обикновена опашка – винаги активираме процеса, блокиран най-рано.

Семафорът е *слаб*, когато контейнера `Q` не е реализиран като обикновена опашка – при изпълнение на `Signal()` активираме процес, който може да не е първи в списъка на чакащите.

Ако пазачът на входа на избирателната секция действа като слаб семафор, може да се получи следната неприятна ситуация:

Първите двама гласоподаватели влизат в секцията, пристига трети гласоподавател (неприятел на пазача) и чака. След него започват да пристигат приятели на пазача и той ги пуска с предимство. Може да се стигне дотам, че третият гласоподавател чака цял ден и гласува последен.

Подобна несправедлива ситуация при достъп до ресурс наричаме *starvation* (гладуване).