Задачи от лекции по операционни системи

Авторът на този файл не гарантира за достоверността му

1. Два процеса p и q. Изпълняват се паралелно. Искаме p2<q2 (p2 да се изпълни преди q2)
   s.init(0)

p	**q**
p1	q1
	s.wait()
p2	q2
s.signal()	
p3	q3

Обяснение:

1. **Инициализация:**
   Семафорът **s** се инициализира със стойност 0. Това означава, че ако процес q достигне **s.wait()**, той ще бъде блокиран, защото стойността на семафора е 0.
2. **Изпълнение на процес p:**
   - p изпълнява първо p1.
   - След това изпълнява p2. Това е критичният участък, който трябва да се изпълни преди q2.
   - Веднага след завършване на p2, процесът p извиква **s.signal()**, като увеличава стойността на семафора от 0 на 1. Това действие уведомява, че p2 е завършил.
3. **Изпълнение на процес q:**
   - q започва с изпълнението на q1.
   - При достигане на **s.wait()**, q се опитва да намали стойността на семафора. Ако все още p не е завършил p2 (и следователно **s.signal()** не е извикан), стойността ще бъде 0 и q ще блокира, докато семафорът не бъде сигнализиран.
   - Когато p завърши p2 и извика **s.signal()**, стойността на семафора се увеличава и q може да премине от wait() към изпълнението на q2.
   - След това q продължава с изпълнението на q3.

2. Два процеса p и q. Изпълняват се паралелно. Искаме p2<q2 за всеки процес q
   s.init(0)

p	**q_a**	**q_b**	**q_c**	.....
p1	q1	q1	...	
	s.wait()	s.wait()		
	s.signal()	s.signal()		
p2	q2	g2	...	
s.signal()				
p3	q3	q3	...	

3. Mutex - p1..pk и q1...qk са критични секции
   Целта е да се гарантира, че никога няма да има едновременен достъп до критичната секция – в даден момент само един от процесите (p или q) може да изпълнява своята критична секция. За тази цел ще използваме **mutex** (взаимно изключване), който се инициализира със стойност 1. За да осигурим взаимно изключение, всеки процес трябва да извика операцията s.wait() (заключване на mutex-а) преди да влезе в своята критична секция и след приключване да извика s.signal() (освобождаване на mutex-а).
   s.init(1)

p	**q**
..	..
..	..
s.wait()	s.wait()
p1	q1
p2	q2
...	..
pk	qk
s.signal()	s.signal()
..	..
..	..

4. Rendezvous – a1<b2 и b1<a2
   sA.init(0)
   sB.init(0)

A	**B**
a1	b1
sA.signal()	sB.signal()
sB.wait()	sA.wait()
a2	b2

   Ако имаме следното пак работи, но не толкова ефективно
   sA.init(0)
   sB.init(0)

A	**B**
a1	b1
sA.signal()	sA.wait()
sB.wait()	sB.signal()
a2	b2

   Ако имаме следното се получава deadlock
   sA.init(0)
   sB.init(0)

A	**B**
a1	b1
sB.wait()	sA.wait()
sA.signal()	sB.signal()
a2	b2

Така как работи:

sA.init(1)

sB.init(0)

A	**B**
a1	b1
sB.wait()	sA.wait()
sA.signal()	sB.signal()
a2	b2