

PCD: Quiz 5

O e-mail do participante (**igor.ribeiro@unifesp.br**) foi registrado durante o envio deste formulário.

Qual o valor do semáforo S quando a thread t termina? *

Semaphore s = new Semaphore(2); // capacity 2			
thread t		thread u	
1	for (int i = 0; i < 10; i++)	4	for (int i = 0; i < 10; i++)
2	{ s.down();	5	{ s.down();
3	s.up(); }	6	s.up(); }

- ☐ 1
- ☒ 2
- ☐ 1 ou 2
- ☐ 0, 1 ou 2

Resposta correta

- ☒ 1 ou 2



A definição clássica de semáforo diz que este é um(a) _____ para resolver o problema da seção crítica. *

- ☐ recurso de hardware para um sistema
- ☒ variável inteira
- ☐ variável inteira associada a um grupo de processos

Resposta correta

- ☒ variável inteira associada a um grupo de processos

O código que altera o valor do semáforo é _____ *

- ☐ protocolo de entrada/saída de um programa
- ☐ protocolo de entrada/saída da seção não crítica
- ☒ protocolo de entrada/saída da seção crítica
- ☐ nenhum dos mencionados



Qual o valor do semáforo S quando a thread t termina? *

Semaphore s = new Semaphore(1); // capacity 1			
thread t		thread u	
1	for (int i = 0; i < 10; i++)	4	for (int i = 0; i < 10; i++)
2	{ s.down();	5	{ s.down();
3	s.up(); }	6	s.up(); }

- ☒ 1
- ☐ 2
- ☐ 1 ou 2
- ☐ 0, 1 ou 2

Considere que as operações de lock() e unlock() do pseudo-código abaixo possuem o funcionamento semelhante ao dos semáforos binários. O que o programa abaixo imprime? *

Lock one = new Lock(); Lock two = new Lock();			
thread t		thread u	
1	one.lock();	6	one.lock();
2	two.lock();	7	two.lock();
3	System.out.println("t");	8	System.out.println("u");
4	two.unlock();	9	two.unlock();
5	one.unlock();	10	one.unlock();

- ☐ "t" depois "u"
- ☒ "t" depois "u" ou "u" depois "t"
- ☐ "u" depois "t"
- ☐ "t" depois "u", "u" depois "t", ou pode não imprimir nada



O programa a seguir consiste em 3 processos simultâneos e 3 semáforos binários. Os semáforos são inicializados como $S0 = 1$, $S1 = 0$, $S2 = 0$. Quantas vezes P0 imprimirá '0'?

*

```
Process P0
while(true)
{
    wait(S0);
    print '0';
    release(S1);
    release(S2);
}
```

```
Process P1
wait(S1);
release(S0);
```

```
Process P2
wait(S2);
release(S0);
```

- ☒ Pelo menos duas vezes
- ☐ Exatamente duas vezes
- ☐ Exatamente três vezes
- ☐ Exatamente uma vez



Considere o código abaixo para os processos P_i , $i = 0, 1, 2, 3, \dots, 9$. O código para P_{10} é idêntico, exceto que usa V (mutex) em vez de P (mutex). Qual é o maior número de processos que podem estar dentro da seção crítica a qualquer momento (o mutex sendo inicializado em 1)? *

```
repeat
  P(mutex)
  {Critical Section}
  V(mutex)
forever
```

- ☐ 1
- ☒ 2
- ☐ 3
- ☐ Nenhuma das anteriores.

Resposta correta

- ☒ Nenhuma das anteriores.



Considere que dois processos, P1 e P2, desejam acessar uma seção crítica do código e utilizam a construção de sincronização mostrada no código abaixo. As variáveis w1 e w2 são compartilhadas e inicializadas como falsas. Qual das seguintes afirmações é VERDADEIRA sobre a construção de código mostrada? *

```
Process P1 :  
while(true)  
{  
  w1 = true;  
  while(w2 == true);  
  Critical section  
  w1 = false;  
}  
Remainder Section  
  
Process P2 :  
while(true)  
{  
  w2 = true;  
  while(w1 == true);  
  Critical section  
  w2 = false;  
}  
Remainder Section
```

- ☒ Não garante exclusão mútua
- ☐ Não garante espera limitada (não é livre de "starvation")
- ☐ Requer que os processos entrem na seção crítica em estrita alternância
- ☐ Não evita deadlock, mas garante exclusão mútua

Resposta correta

- ☒ Não evita deadlock, mas garante exclusão mútua

Este formulário foi criado em Universidade Federal de Sao Paulo.

Google Formulários



