|  | UNIVERSIDADE FEDERAL DE SERGIPE  CAMPUS ITABAIANA  DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO  SINF0072 - SISTEMAS OPERACIONAIS PROF: ANDRÉ LUIS MENESES SILVA |
| --- | --- |

**3. SINCRONIZAÇÃO**

1. Operações com semáforos

A seguir, apresentamos uma sequência de operações do semáforo no início e no final das tarefas A, B, C. Considere que cada tarefa executa em um núcleo de processador dedicado. E considere que cada ação (P(Sx), V(Sx) ou .) possui tempo igual a 1T.

|  | Task A | Task B | Task C |
| --- | --- | --- | --- |
| 1 | P(SA) | P(SB) | P(SC) |
| 2 | P(SA) | . | P(SC) |
| 3 | P(SA) | . | P(SC) |
| 4 | . | . | . |
| 5 | . | . | . |
| 6 | . | V(SC) | V(SB) |
| 7 | V(SB) | V(SA) | V(SB) |
| 8 | END | . | V(SA) |
| 9 |  | END | END |

Determine para os 6 casos a,b,c,d,e,f apresentados na tabela abaixo, se e em qual sequência as tarefas são executadas, usando as inicializações das variáveis do semáforo dadas na tabela.

| Semáforos | a) | b) | c) | d) | e) | f) | g) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SA | 2 | 3 | 2 | 0 | 3 | 1 | 1 |
| SB | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| SC | 2 | 2 | 1 | 3 | 3 | 3 | 1 |

1. Deadlock, nenhuma task finaliza, TA e TC bloqueadas no T3 (tempo 3) e TB bloqueado em T1 (tempo 1).

| Semáforos | a) |
| --- | --- |
| SA | 2 1 0 |
| SB | 0 |
| SC | 2 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | X |  |  |  |  |  |  |  |  |  |  |
| TB | X | X | X |  |  |  |  |  |  |  |  |  |  |
| TC | P(SC) | P(SC) | X |  |  |  |  |  |  |  |  |  |  |

1. Não existe Deadlock, Todas as tasks finalizam. TA em T8(tempo 8), TB em T16(tempo 16) e TC em T20 (tempo 20). Com isso concluímos que qualquer teste que tenha no mínimo SA, SB e SC como 3,0 e 2 respectivamente, não haverá deadlock.

| Semáforos | b) |
| --- | --- |
| SA | 3 2 1 2 |
| SB | 0 1 0 |
| SC | 2 1 0 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | **…T12** | T13 | T14 | T15 | T16 | T17 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | P(SA) | - | - | - | V(SB) | END |  |  |  |  |  |  |
| TB | X | X | X | X | X | X | X | P(SB) | - | V(SC) | V(SA) | - | END |  |
| TC | P(SC) | P(SC) | X | X | X | X | X | X | X | X | P(SC) | - | - | V(SB) |

1. Deadlock pois tarefa C não executa. TA finaliza em T3, TB finaliza em T9.

| Semáforos | c) |
| --- | --- |
| SA | 2 1 0 1 0 |
| SB | 1 0 1 |
| SC | 1 0 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | X | X | X | X | X | P(SA) | - | - | - | V(SB) | END |
| TB | P(SB) | - | - | - | - | V(SC) | V(SA) | - | END |  |  |  |  |
| TC | P(SC) | X | X | X | X | X | P(SC) | X | X | X | X | X |  |

1. Exite Deadlock, a tarefa TA não executa e fica bloqueada em T15. TB finaliza em T15 e TC finaliza em T9

| Semáforos | d) |
| --- | --- |
| SA | 0 1 0 1 |
| SB | 0 1 0 1 |
| SC | 3 2 1 0 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 | T15 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | X | X | X | X | X | X | X | X | P(SA) | X | X | X | X | P(SA) | X |
| TB | X | X | X | X | X | X | P(SB) | - | - | - | - | V(SC) | V(SA) | - | END |
| TC | P(SC) | P(SC) | P(SC) | - | - | V(SB) | V(SB) | V(SA) | END |  |  |  |  |  |  |

1. Não tem deadlock, todas as task são executadas sem problemas. TA finaliza em T8. TB e TC finalizam em T9.

| Semáforos | e) |
| --- | --- |
| SA | 3 2 1 0 |
| SB | 1 0 1 2 |
| SC | 3 2 1 0 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | P(SA) | - | - | - | V(SB) | END |  |  |  |  |  |  |
| TB | P(SB) | - | - | - | - | V(SC) | V(SA) | - | END |  |  |  |  |  |
| TC | P(SC) | P(SC) | P(SC) | - | - | V(SB) | V(SB) | V(SA) | END |  |  |  |  |  |

1. Não tem deadlock, todas as task são executadas sem problemas. TA finaliza em T20. TB finaliza em T15. TC finaliza em T9

| Semáforos | f) |
| --- | --- |
| SA | 1 0 1 0 1 |
| SB | 0 1 0 1 |
| SC | 3 2 1 0 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 | T15 | T16 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | X | X | X | X | X | X | X | P(SA) | X | X | X | X | P(SA) | - | - |
| TB | X | X | X | X | X | X | P(SB) | - | - | - | - | V(SC) | V(SA) | - | END |  |
| TC | P(SC) | P(SC) | P(SC) | - | - | V(SB) | V(SB) | V(SA) | END |  |  |  |  |  |  |  |

| T17 | T18 | T19 | T20 |
| --- | --- | --- | --- |
| - | - | V(SB) | END |
|  |  |  |  |
|  |  |  |  |

1. Possui deadlock. Apenas de TB finaliza, em T9. TA fica bloqueado em T9 e TC em T8.

| Semáforos | g) |
| --- | --- |
| SA | 1 0 1 |
| SB | 1 0 |
| SC | 1 0 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | X | X | X | X | X | X | P(SA) | X |  |  |  |  |  |
| TB | P(SB) | - | - | - | - | V(SC) | V(SA) | - | END |  |  |  |  |  |
| TC | P(SC) | X | X | X | X | X | P(SC) | X |  |  |  |  |  |  |

|  | UNIVERSIDADE FEDERAL DE SERGIPE  CAMPUS ITABAIANA  DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO  SINF0072 - SISTEMAS OPERACIONAIS PROF: ANDRÉ LUIS MENESES SILVA |
| --- | --- |

A seguir, apresentamos uma nova sequência de operações do semáforo no início e no final das tarefas A, B, C. Considere que cada tarefa executa em um núcleo de processador dedicado. E considere que cada ação (P(Sx), V(Sx) ou .) possui tempo igual a 1T.

|  | Task A | Task B | Task C |
| --- | --- | --- | --- |
| 1 | P(SA) | P(SB) | P(SC) |
| 2 | P(SA) | P(SA) | P(SC) |
| 3 | V(SA) | . | P(SB) |
| 4 | . | . | . |
| 5 | . | . | . |
| 6 | . | P(SC) | V(SB) |
| 7 | V(SC) | V(SA) | V(SB) |
| 8 | END | END | V(SA) |
| 9 |  |  | END |

Determine para os 6 casos a,b,c,d,e,f apresentados na tabela abaixo, se e em qual sequência as tarefas são executadas, usando as inicializações das variáveis do semáforo dadas na tabela.

| Semáforos | a) | b) | c) | d) | e) | f) | g) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SA | 2 | 1 | 2 | 0 | 3 | 2 | 1 |
| SB | 0 | 0 | 1 | 0 | 1 | 2 | 1 |
| SC | 2 | 1 | 1 | 2 | 1 | 2 | 1 |

1. Há deadlock, apenas a tarefa A executa em T8. TB fica bloqueado em T1 e TC em T3

| Semáforos | A) |
| --- | --- |
| SA | 2 1 0 1 |
| SB | 0 |
| SC | 2 1 0 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | V(SA) | - | - | - | V(SC) | END |  |  |  |  |  |
| TB | X | X | X | X | X | X | x | x |  |  |  |  |  |
| TC | P(SC) | P(SC) | X | X | X | X | x | x |  |  |  |  |  |

1. Há deadlock. Nenhuma tarefa executa.

| Semáforos | B) |
| --- | --- |
| SA | 1 |
| SB | 0 |
| SC | 1 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | X |  |  |  |  |  |  |  |  |  |  |  |
| TB | X | X |  |  |  |  |  |  |  |  |  |  |  |
| TC | P(SC) | X |  |  |  |  |  |  |  |  |  |  |  |

1. Há deadlock. TA executa em T8 e TB em T10, mas TC permanece bloqueado em T2.

| Semáforos | C) |
| --- | --- |
| SA | 2 1 0 1 0 1 |
| SB | 1 0 |
| SC | 1 0 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | V(SA) | - | - | - | V(SC) | END |  |  |  |  |  |
| TB | P(SB) | X | X | P(SA) | - | - | - | P(SC) | V(SA) | END |  |  |  |
| TC | P(SC) | X | X | X | X | X | X | X | X |  |  |  |  |

1. Há deadlock. Nenhuma tarefa executa. TA e TB ficam bloqueadas em T1 e TC em T3

| Semáforos | D) |
| --- | --- |
| SA | 0 |
| SB | 0 |
| SC | 2 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | X | X | X |  |  |  |  |  |  |  |  |  |  |
| TB | X | X | X |  |  |  |  |  |  |  |  |  |  |
| TC | P(SC) | P(SC) | X |  |  |  |  |  |  |  |  |  |  |

1. Há deadlock. TA finaliza em T\* e TB em T10, mas TC fica vloqueado em T2.

| Semáforos | E) |
| --- | --- |
| SA | 3 2 1 0 1 2 |
| SB | 1 0 |
| SC | 1 0 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | V(SA) | - | - | - | V(SC) | END |  |  |  |  |  |
| TB | P(SB) | P(SA) | - | - | - | X | X | P(SC) | V(SA) | END |  |  |  |
| TC | P(SC) | X | X | X | X | X | X | X |  |  |  |  |  |

1. Não tem deadlock.

| Semáforos | F) |
| --- | --- |
| SA | 2 1 0 1 0 1 |
| SB | 2 1 0 1 2 |
| SC | 2 1 0 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | P(SA) | V(SA) | - | - | - | V(SC) | END |  |  |  |  |  |
| TB | P(SB) | X | X | P(SA) | - | - | - | P(SC) | V(SA) | END |  |  |  |
| TC | P(SC) | P(SC) | P(SB) | - | - | V(SB) | V(SB) | V(SA) | END |  |  |  |  |

1. Tem deadlock, todas as tarefas são bloqueadas em T2

| Semáforos | G) |
| --- | --- |
| SA | 1 0 |
| SB | 1 0 |
| SC | 1 0 |

| Task | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TA | P(SA) | X |  |  |  |  |  |  |  |  |  |  |  |
| TB | P(SB) | X |  |  |  |  |  |  |  |  |  |  |  |
| TC | P(SC) | X |  |  |  |  |  |  |  |  |  |  |  |

|  | UNIVERSIDADE FEDERAL DE SERGIPE  CAMPUS ITABAIANA  DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO  SINF0072 - SISTEMAS OPERACIONAIS PROF: ANDRÉ LUIS MENESES SILVA |
| --- | --- |

**2. Códigos em Python.**

**Nos seguintes códigos explique o comportamento do código e o conteúdo que será exibido ao final de sua execução.**

| **#A**  **from threading import \***  **import time**  **l=Lock()**  **def wish(name,age):**  **for i in range(3):**  **l.acquire()**  **print("Hi",name)**  **time.sleep(2)**  **print("Your age is",age)**  **l.release()**  **t1=Thread(target=wish, args=("Sireesh",15)) t2=Thread(target=wish, args=("Nitya",20)) t1.start()**  **t2.start()** |
| --- |
| **#B**  **from threading import \***  **import time**  **s=Semaphore(2)**  **def wish(name,age):**  **for i in range(3):**  **s.acquire()**  **print("Hi",name)**  **time.sleep(2)**  **s.release()**  **t1=Thread(target=wish, args=("Sireesh",15)) t2=Thread(target=wish, args=("Nitya",20)) t3=Thread(target=wish, args=("Shiva",16)) t4=Thread(target=wish, args=("Ajay",25)) t1.start()**  **t2.start()**  **t3.start()**  **t4.start()** |

**SAÍDA A:**

**Hi Sireesh**

**Your age is 15**

**Hi Sireesh**

**Your age is 15**

**Hi Sireesh**

**Your age is 15**

**Hi Nitya**

**Your age is 20**

**Hi Nitya**

**Your age is 20**

**Hi Nitya**

**Your age is 20**

1. **Neste exemplo é utilizado o Lock, ele irá bloquear o ‘recurso’ em uma thread e a próxima thread só irá ser executada quando a primeira thread que possui o recurso finalizar, dessa forma, liberando o recurso para a próxima thread. A intenção do Lock é justamente simular essa necessidade do recurso e um programa em espera.**

**SAÍDA B:**

**Hi SireeshHi**

**Nitya**

**HiHi SireeshNitya**

**Hi Sireesh**

**Hi Nitya**

**Hi Shiva**

**Hi Ajay**

**Hi Shiva**

**Hi Ajay**

**Hi Shiva**

**Hi Ajay**

1. **O código traz o uso de semáforos, as 4 threads são inicializadas com os parâmetros, porém as tarefas possuem apenas 2 contadores para permitir que apenas 2 threads sejam executadas simultaneamente.**

|  | UNIVERSIDADE FEDERAL DE SERGIPE  CAMPUS ITABAIANA  DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO  SINF0072 - SISTEMAS OPERACIONAIS PROF: ANDRÉ LUIS MENESES SILVA | |
| --- | --- | --- |
|  |  |  |
| **#C**  **from threading import Lock, Thread**  **lock = Lock()**  **g = 0**  **def add\_one():**  **global g**  **lock.acquire()**  **g += 1**  **lock.release()**  **def add\_two():**  **global g**  **lock.acquire()**  **g += 2**  **lock.release()**  **threads = []**  **for func in [add\_one, add\_two, add\_two, add\_one, add\_one, add\_two]:**  **threads.append(Thread(target=func))**  **threads[-1].start()**  **for thread in threads:**  **thread.join()**  **print(g)** | |  |

S**AÍDA C:**

**9**

1. **O código acima modifica a mesma variável, a utilização do Lock é necessária para não haver confusões na execução pois sem o Lock, as threads iriam modificar a mesma variável podendo ter inconsistências no valor final.**

|  | UNIVERSIDADE FEDERAL DE SERGIPE  CAMPUS ITABAIANA  DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO  SINF0072 - SISTEMAS OPERACIONAIS PROF: ANDRÉ LUIS MENESES SILVA |
| --- | --- |

**3. Resolvendo problemas com Sincronização**

1. A seguir é apresentado trecho de código Python. Análise o código e responda as seguintes questões:
2. Explique a finalidade do código apresentado?
3. Qual o resultado após execução do código?
4. Execute o código 10 vezes. Os resultados foram iguais? Caso negativo, por qual motivo?
5. Utilize mecanismos de sincronização de forma que ao final da execução do código conta2 possua saldo 100 e conta1 possua saldo 0.

**import threading**

**import time**

**class ContaBancaria():**

**def \_\_init\_\_(self, nome, saldo):**

**self.nome = nome**

**self.saldo = saldo**

**def \_\_str\_\_(self):**

**return self.nome**

**conta1 = ContaBancaria("conta1", 100)**

**conta2 = ContaBancaria("conta2", 0)**

**class ThreadTransferenciaEntreContas(threading.Thread):**

**def \_\_init\_\_(self, origem, destino, valor):**

**threading.Thread.\_\_init\_\_(self)**

**self.origem = origem**

**self.destino = destino**

**self.valor = valor**

**def run(self):**

**origem\_saldo\_inicial = self.origem.saldo**

**origem\_saldo\_inicial -= self.valor**

**time.sleep(0.001)**

**self.origem.saldo = origem\_saldo\_inicial**

**destino\_saldo\_inicial = self.destino.saldo**

**destino\_saldo\_inicial += self.valor**

**time.sleep(0.001)**

**self.destino.saldo = destino\_saldo\_inicial**

**if \_\_name\_\_ == "\_\_main\_\_":**

**threads = []**

**for i in range(100):**

**threads.append(ThreadTransferenciaEntreContas(conta1, conta2, 1))**

**for thread in threads:**

**thread.start()**

**for thread in threads:**

**thread.join()**

**print('Saldo da', conta1, ':', conta1.saldo)**

**print('Saldo da', conta2, ':', conta2.saldo)**

1. **O código simula uma transferência entre duas contas bancárias por meio da utilização de threads, elas executam e modificam os saldos das contas.**
2. **A execução mostra que foi passado algum valor da conta 1 para a 2, mas o valor passado não está bem definido, visto que as threads executam em paralelo e modificam os mesmos valores, causando inconsistências nos valores.**
3. **Sempre os resultados são diferentes, isso se dá pela execução paralela das thread e pela utilização do mesmo valor em processos diferentes o que normalmente pode trazer inconsistências no valor final.**
4. **Correção: Uso do Lock no método run() resolve e faz com o que a conta 1 zere e a conta dois fique com 100.**

**from threading import Thread, Lock**

**import time**

**class ContaBancaria():**

**def \_\_init\_\_(self, nome, saldo):**

**self.nome = nome**

**self.saldo = saldo**

**def \_\_str\_\_(self):**

**return self.nome**

**conta1 = ContaBancaria("conta1", 100)**

**conta2 = ContaBancaria("conta2", 0)**

**l=Lock()**

**class ThreadTransferenciaEntreContas(Thread):**

**def \_\_init\_\_(self, origem, destino, valor):**

**Thread.\_\_init\_\_(self)**

**self.origem = origem**

**self.destino = destino**

**self.valor = valor**

**def run(self):**

**l.acquire()**

**origem\_saldo\_inicial = self.origem.saldo**

**origem\_saldo\_inicial -= self.valor**

**time.sleep(0.001)**

**self.origem.saldo = origem\_saldo\_inicial**

**destino\_saldo\_inicial = self.destino.saldo**

**destino\_saldo\_inicial += self.valor**

**time.sleep(0.001)**

**self.destino.saldo = destino\_saldo\_inicial**

**l.release()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**threads = []**

**for i in range(100):**

**threads.append(ThreadTransferenciaEntreContas(conta1, conta2, 1))**

**for thread in threads:**

**thread.start()**

**for thread in threads:**

**thread.join()**

**print('Saldo da', conta1, ':', conta1.saldo)**

**print('Saldo da', conta2, ':', conta2.saldo)**