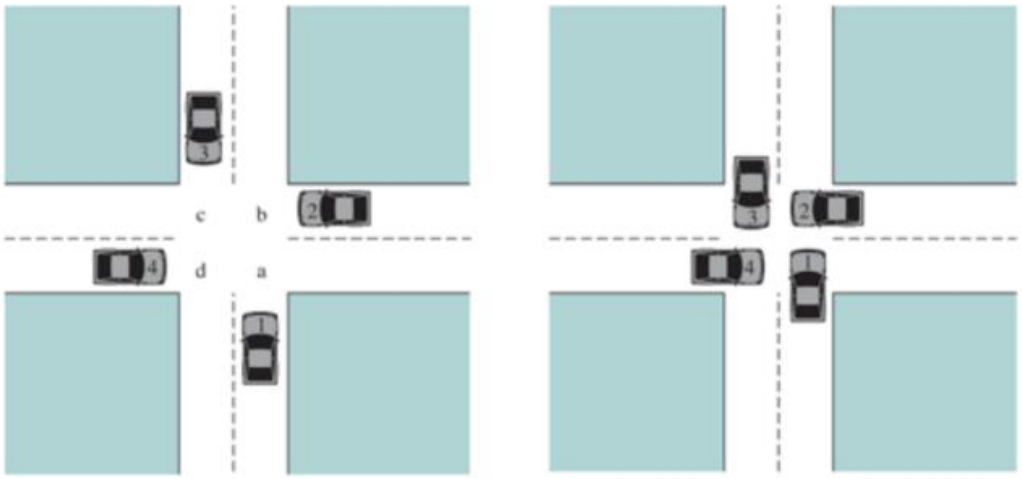
Natalia Ramirez

TAREA DEADLOCK

6.1



Exclusion mutua: por las calles solo puede pasar un carro a la vez. Cada carro esta ocupando una calle/cuadrante.

Retencion y espera: cada carro esta reteniendo una calle/cuadrante mientras espera por otro para poder cruzar.

No preemption: Ninguno de los recursos puede ser pausado o detenido.

Espera circular: hay un lazo de espera para las necesidades de los carros. El carro 1 espera que el 2 cruce, el 2 espera que el 3 cruce, el 3 espera que el 4 cruce y el 4 espera que el 1 cruce.

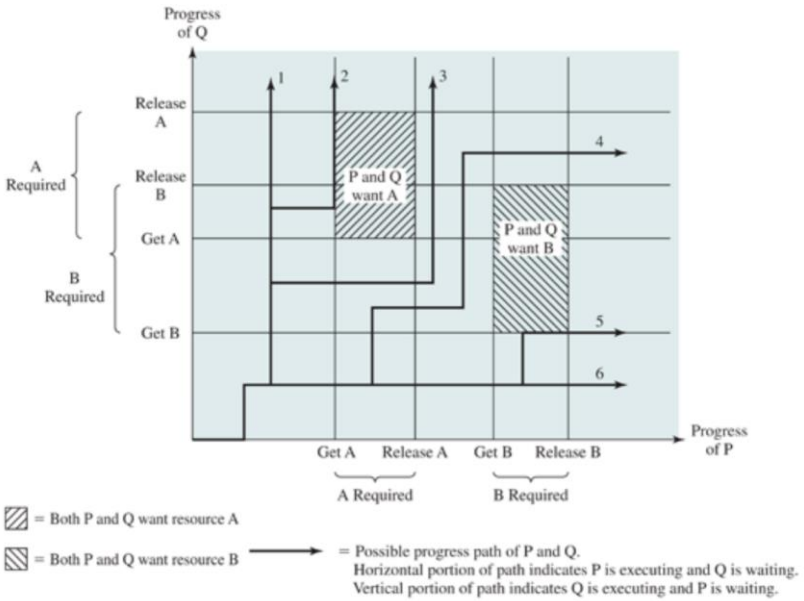
6.2

Prevención: Asegurarse que no lleguen carros a la intersección si no esta libre su cuadrante necesario.

Detección: detectar que hay un cuadrante ocupado que es necesitado por otro carro.

Evitación: reservar ciertos cuadrantes o bloquearlos según los carros quieran cruzar o acercarse, como el funcionamiento de un semáforo.

6.3



1. Q adquiere a B y A, entonces libera B y A. Cuando P reanuda su ejecución, este adquiere ambos recursos.
2. Q adquiere B y A. P ejecuta y se bloquea sobre una petición para A. Q libera a B y A. Cuando P reanuda su ejecución, este adquiere ambos recursos.
3. P requiere a A y despues Q adquiere a A y libera a B y despues a A.
4. Q Obtiene a B y luego lo libera. Entonces P adquiere a A para despues Q obtener a A y luego liberar a B. Por último P aquiere a B.
5. Q adquiere a B y luego P requiere a B para luego liberarlo
6. P libera a B

6.4

En el caso presentado en la figura no puede ocurrir un deadlock ya que P no requiere ambos recursos al mismo tiempo en ninguno de los caminos. Esto significa que, aunque hay competencia de los recursos entre P y Q, ninguno queda bloqueado por el otro y viceversa. No se hace una espera circular.

6.5

4 resource types: A (15 instances); B (6 instances)

C (9 instances); D (10 instances)

Natalia Ramirez

Available			
A	B	C	D
6	3	5	4

	Current allocation				Maximum demand			
Process	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

a)

asignados A = 2+0+4+1+1+1 = 9

totales A = 15

disponibles A = 15 – 9 = 6

asignados B = 3

totales B = 6

disponibles B = 6 – 3 = 6

asignados C = 4

totales C= 9

disponibles C = 9 – 4 = 5

asignados D = 6

totales D = 10

disponibles D = 10 – 6 = 4

b) NEED MATRIX

PROCESS	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	3	4	3	3

c)

P3 TERMINA

A	B	C	D
7	3	5	5

P1 TERMINA

A	B	C	D
7	4	6	6

P2 TERMINA

A	B	C	D
11	5	6	8

P0 TERMINA

A	B	C	D
13	5	8	9

P4 TERMINA

A	B	C	D
14	6	8	9

P5 TERMINA

A	B	C	D
15	6	9	10

d)

NEED MATRIX

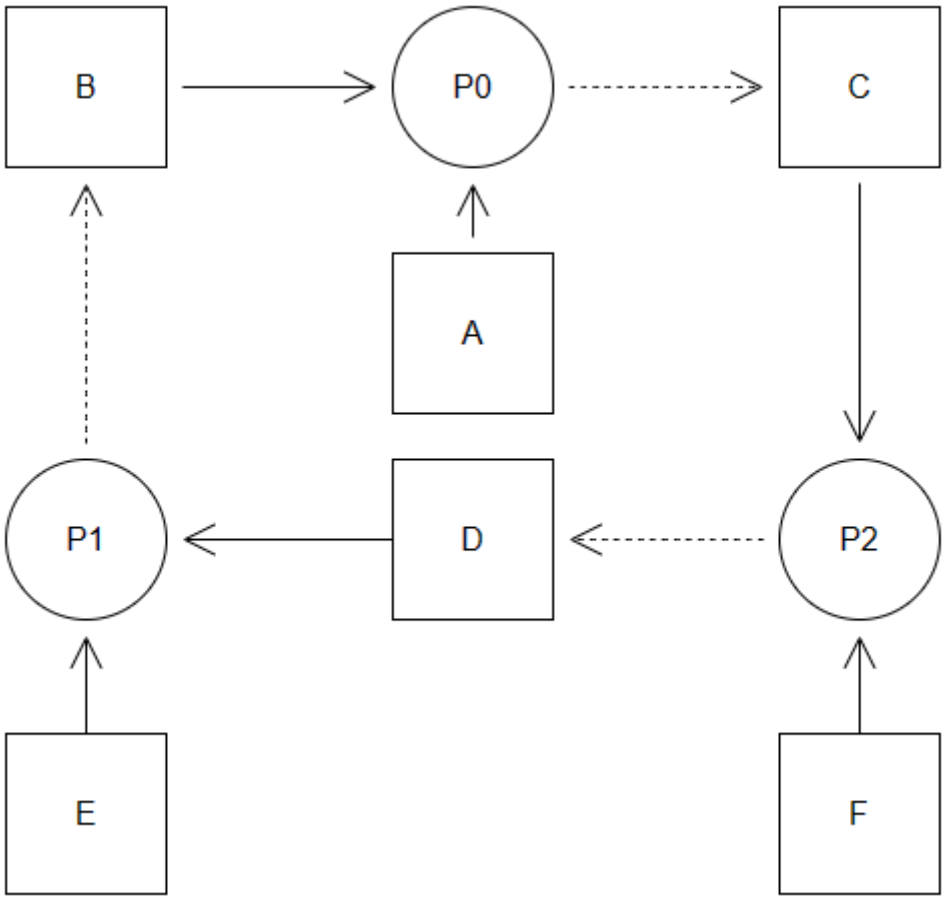
PROCESS	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	0	2	0	0

Este requerimiento no debería ser dado ya que resulta en interbloqueo entre los procesos P2 y P0 por el recurso B. La secuencia hasta llegar a ese estado es: P5, P4, P3, P1

6.6

a)

Natalia Ramirez



b)

P2: get D, get C, get F

P1: get E, get D, get B

P0: get C, get B, get A

Natalia Ramirez