Sistemas Operativos

Semáforos

Rodrigo Urrea

P1.

Baño compartido

Problema del baño compartido

Un estadio posee un **único baño** que debe ser compartido por hinchas **rojos** y **azules**. El baño es amplio y admite un número ilimitado de personas. El problema consiste en **evitar** que los **hinchas rojos** se encuentren con los **hinchas azules** dentro del baño.

Los hinchas rojos solicitan entrar al baño invocando ***entrar(ROJO)*** y notifican su salida con ***salir(ROJO)***, mientras que los hinchas azules invocan ***entrar(AZUL)*** y ***salir(AZUL)***.

Implementación incorrecta

```
enum { ROJO = 0, AZUL = 1 };

int mutex = 0;
// Este mutex representa el acceso al baño.
// El equipo que lo tiene es el que está adentro.
int cantidad[2] = {0, 0};

void entrar(int color) {
    if (cantidad[color] == 0) {
        while (mutex)
            ;
        mutex = 1;
    }
    cantidad[color]++;
}

void salir(int color) {
    cantidad[color]--;
    if (cantidad[color] == 0) {
        mutex = 0;
    }
}
```

P1.a Muestre mediante un diagrama de threads que la implementación anterior no garantiza exclusión mutua.

	entrar(**AZUL**)
entrar(**ROJO**)	

Variables Globales:
mutex = 0
cantidad[ROJO] = 0
cantidad[AZUL] = 0

P1.a Muestre mediante un diagrama de threads que la implementación anterior no garantiza exclusión mutua

	entrar(**AZUL**)
entrar(**ROJO**)	
	if(cantidad[**AZUL**] == 0)
if(cantidad[**ROJO**] == 0)	

Variables Globales:
mutex = 0
cantidad[ROJO] = 0
cantidad[AZUL] = 0

P1.a Muestre mediante un diagrama de threads que la implementación anterior no garantiza exclusión mutua

	entrar(**AZUL**)
entrar(**ROJO**)	
	if(cantidad[**AZUL**] == 0)
if(cantidad[**ROJO**] == 0)	mutex == 0
mutex == 0	

Variables Globales:
mutex = 0
cantidad[ROJO] = 0
cantidad[AZUL] = 0

P1.a Muestre mediante un diagrama de threads que la implementación anterior no garantiza exclusión mutua

	entrar(**AZUL**)
entrar(**ROJO**)	
	if(cantidad[**AZUL**] == 0)
if(cantidad[**ROJO**] == 0)	mutex == 0
mutex == 0	
	mutex = 1
mutex = 1	

Variables Globales:
mutex = 1
cantidad[ROJO] = 0
cantidad[AZUL] = 0

P1.a Muestre mediante un diagrama de threads que la implementación anterior no garantiza exclusión mutua

	entrar(**AZUL**)
entrar(**ROJO**)	
	if(cantidad[**AZUL**] == 0)
if(cantidad[**ROJO**] == 0)	mutex == 0
mutex == 0	
	mutex = 1
mutex = 1	cantidad[**AZUL**]++
cantidad[**ROJO**]++	

Variables Globales:
mutex = 1
cantidad[ROJO] = 1
cantidad[AZUL] = 1

P1.b Corregir solución incorrecta

Escriba una solución **correcta** y **eficiente** para este problema utilizando 3 semáforos. No importa si en su solución algunos procesos sufren "***hambruna***".

Hint: utilice la estructura de la solución incorrecta.

correcta: no hay **data races**.

eficiente: no hay **busy waiting**.

P2.

Baño compartido sin hambruna

Problema del baño compartido

Considere ahora una solución en la que no se produzca **hambruna**. Para lograr esto es necesario que ningún hincha entre al baño **mientras haya hinchas del otro equipo esperando**. Luego, cuando sale el último hincha del baño, **entran todos los hinchas del equipo contrario** que estaban **esperando**.

Por ejemplo, si hay dos hinchas del equipo rojo en el baño y un hincha azul en espera, el siguiente hincha rojo en llegar no podrá entrar hasta que haya entrado (y salido) el azul.

Implementación incorrecta

```c
#include <semaphore.h>
enum { ROJO = 0, AZUL = 1 };

sem_t mutex;  // sem_init(&mutex, 0, 1);
sem_t sem[2]; // sem_init(&sem[ROJO], 0, 1);
              // sem_init(&sem[AZUL], 0, 1);
int esperan[2] = {0, 0};
int adentro[2] = {0, 0};

void entrar(int color) {
    // el oponente del equipo AZUL es el equipo ROJO y viceversa.
    int oponente = !color;

    sem_wait(&mutex);

    // Si hay hinchas del otro equipo en el baño o en la cola,
    // se debe esperar.
    if (adentro[oponente] > 0 || esperan[oponente] > 0) {
        esperan[color]++;
        sem_post(&mutex);
        sem_wait(&sem[color]); // se pone el thread en espera
        sem_wait(&mutex);
    }
    adentro[color]++;
    sem_post(&mutex);
}
```

```c
void salir(int color) {
    int oponente = !color;

    sem_wait(&mutex);

    adentro[color]--;
    if (adentro[color] == 0) {
        for (int i = 0; i < esperan[oponente]; i++)
            sem_post(&sem[oponente]);
        esperan[oponente] = 0;
    }

    sem_wait(&mutex);
}
```

P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.

ROJO 1	AZUL	ROJO 2
entrar(**ROJO**)		
salir(**ROJO**)		

P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.

ROJO 1	AZUL	ROJO 2
entrar(**ROJO**)		
salir(**ROJO**)	entrar(**AZUL**)	entrar(**ROJO**)

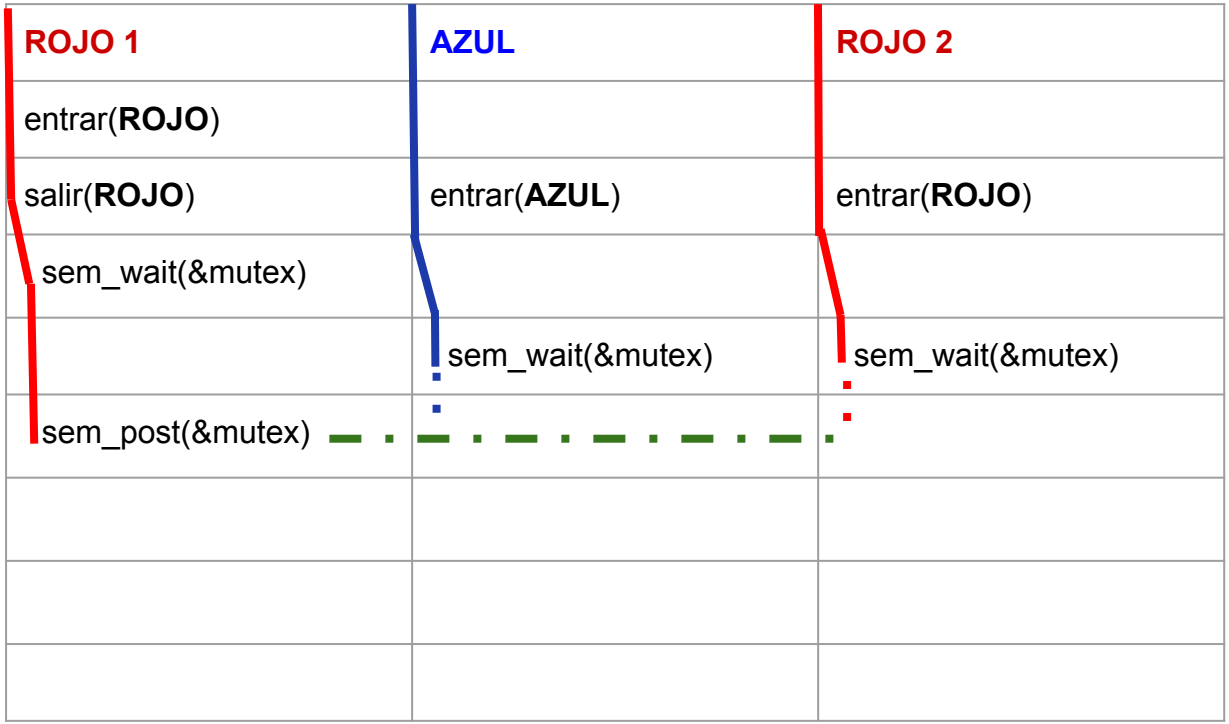
P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.

ROJO 1	AZUL	ROJO 2
entrar(**ROJO**)		
salir(**ROJO**)	entrar(**AZUL**)	entrar(**ROJO**)
sem_wait(&mutex)		

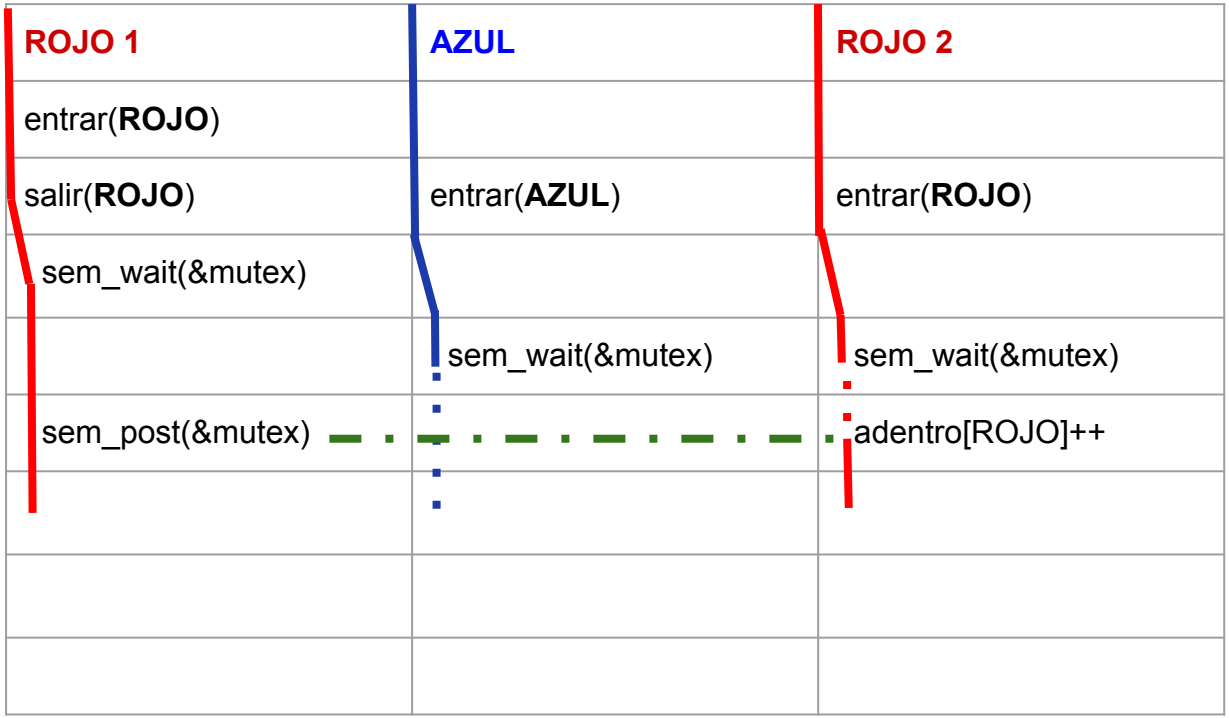
P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.

ROJO 1	AZUL	ROJO 2
entrar(**ROJO**)		
salir(**ROJO**)	entrar(**AZUL**)	entrar(**ROJO**)
sem_wait(&mutex)		
	sem_wait(&mutex)	sem_wait(&mutex)

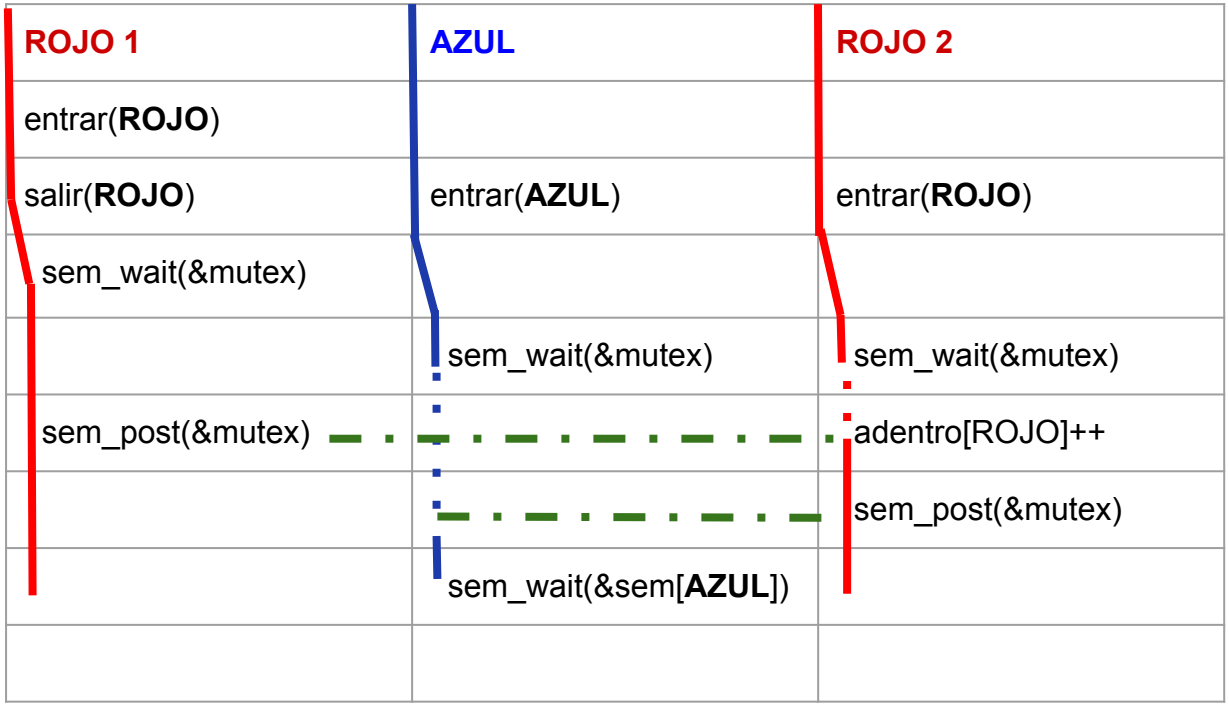
P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.



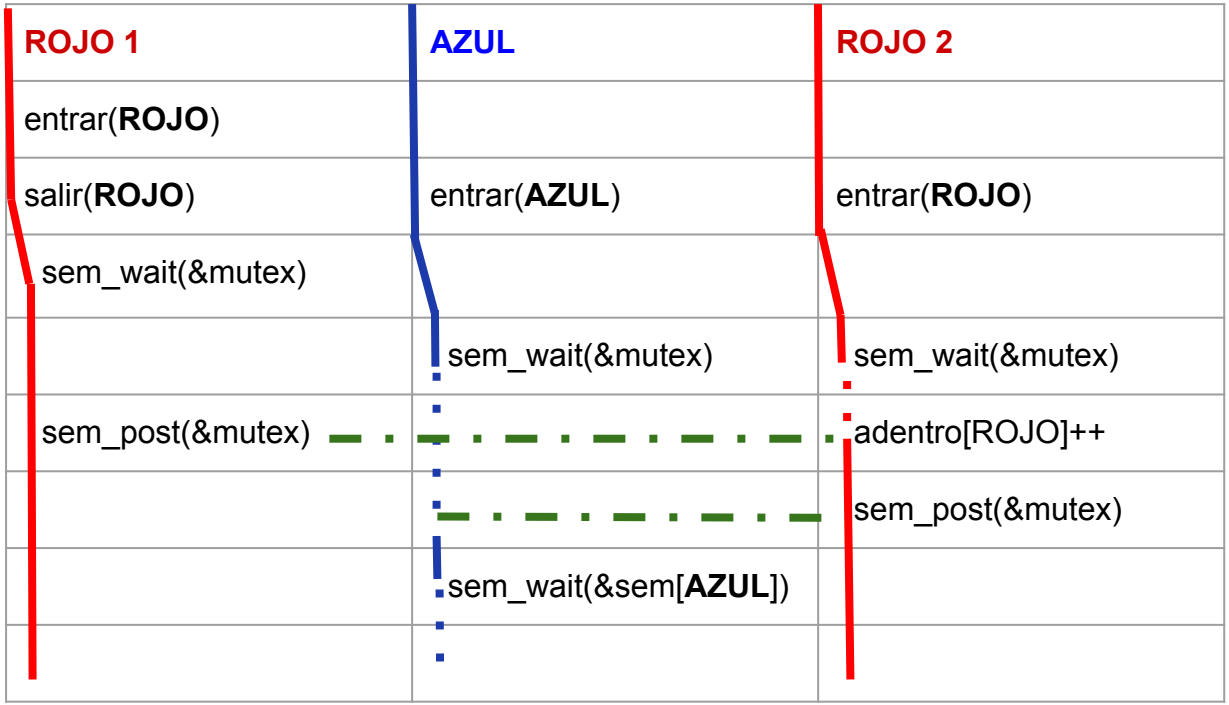
P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.



P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.



P2.a Muestre mediante un diagrama de threads que la implementación anterior no evita hambruna.



P2.b Corregir solución incorrecta

Reprograme la solución anterior de modo que siempre funcione correctamente. Utilice la siguiente metodología:

- Utilice **2 colas FIFO** globales, una para cada equipo.
- Cuando un hincha deba esperar para entrar al baño, cree un **semáforo con 0 tickets** y póngalo en la cola correspondiente. Luego, suspenda el thread solicitando un ticket a este semáforo.
- Cuando salga el último hincha de un equipo y haya hinchas del otro en espera, **extraiga todos los semáforos de esa cola** y deposite en cada uno de ellos un ticket para permitirle a los hinchas en espera entrar al baño.
- Utilice un **semáforo para garantizar exclusión mutua** en el acceso a las variables globales.