Programación Concurrente y Distribuida

Práctica 3 Curso 2023/24 3º Curso

El problema de la exclusión mutua

Introducción

El escenario más representativo de los problemas de concurrencia está compuesto por dos procesos concurrentes que deben acceder en exclusión mutua a su sección crítica. Para comprender la solución propuesta en el *algoritmo de Dekker* es conveniente abordar soluciones previas no correctas que permiten, además, ilustrar situaciones que aparecen con frecuencia en una gran variedad de problemas de concurrencia. Con el objetivo de clarificar dichas situaciones se utilizará una simulación del problema según la Figura 1.

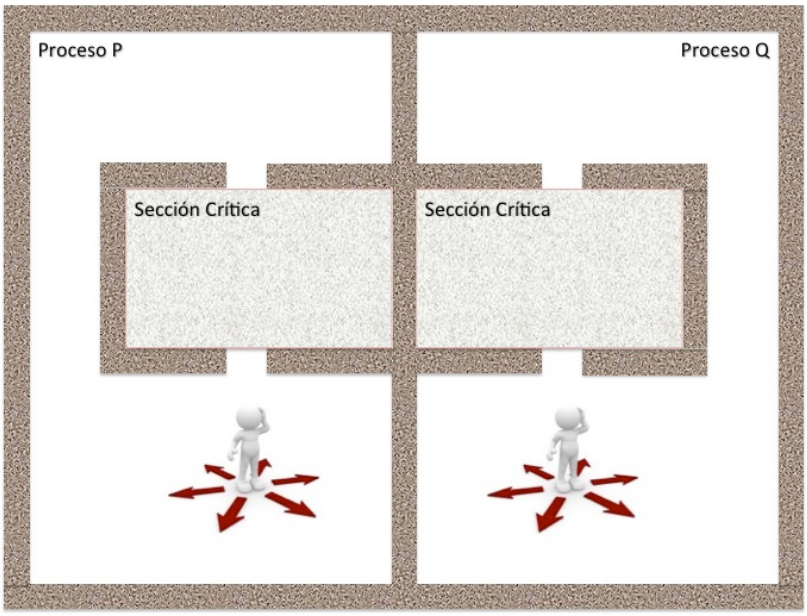

Figura 1. Simulación del problema de la exclusión mutua para dos procesos.

Objetivos

Esta práctica tiene como objetivos fundamentales:

- Comprender la problemática de sincronizar procesos concurrentes.
- Programar procesos concurrentes que ilustren, mediante escenarios, dicha problemática.

Entorno

Los ordenadores del laboratorio ya están preparados para realizar las prácticas. Si el alumno desea realizar las prácticas en su ordenador deberá tener instalado Linux (en el laboratorio está instalada la distribución OpenSuse 12.3 y es sobre la que se deberá garantizar el correcto funcionamiento de las prácticas).

Memoria compartida

Linux incluye tres mecanismos de comunicación entre procesos (*IPCs)*: semáforos, colas de mensajes y memoria compartida, que utilizan las definiciones comunes del fichero de cabecera <*sys/ipc.h*>. En esta práctica se va a utilizar memoria compartida para poder programar los distintos escenarios para ilustrar el problema de la sección crítica.

Para crear una zona de memoria compartida se utiliza la función `shmget` incluida en <*sys/shm.h*>. Para utilizar la memoria compartida creada es necesario vincular dicha zona de memoria a una variable mediante `shmat`. Cuando ya no se desea utilizar la memoria compartida a través de dicha variable, es necesario deshacer el vínculo con `shmdt.` Para liberar la memoria compartida creada con `shmget` se debe invocar la función `shmctl.`

Ejercicio 1 (mini ejemplos):

Consulte el *man* de `shmget, shmat y shmdt`.

Escriba un proceso `p1` que cree una zona de memoria compartida con espacio para un entero y escriba en dicha zona el número que recibe como parámetro.

Ejercicio 2 (mini ejemplos):

Escriba un proceso `p2` que muestre en pantalla el valor de la zona de memoria compartida creada por el proceso `p1` del Ejercicio 1.

Ejercicio 3 (mini ejemplos):

Consulte el *man* de `shmctl.`

Modifique los procesos `p1` y `p2` para que escriban y lean de la memoria compartida en un bucle interactivo (hasta que se pulse una tecla). Escriba un proceso `p3` que libere la zona de memoria compartida. Compruebe qué sucede en función del punto de ejecución en que se encuentren `p1` y `p2.`

Preparando el terreno

Compruebe, recreando la simulación de la Figura 2, cómo los dos procesos pueden violar la exclusión mutua si no se añade ningún código de sincronización. Tenga en cuenta que: i) un proceso puede decidir entrar en su sección crítica en el momento que desee y todas las veces que desee; ii) un proceso no puede permanecer en su sección crítica un tiempo indefinido.

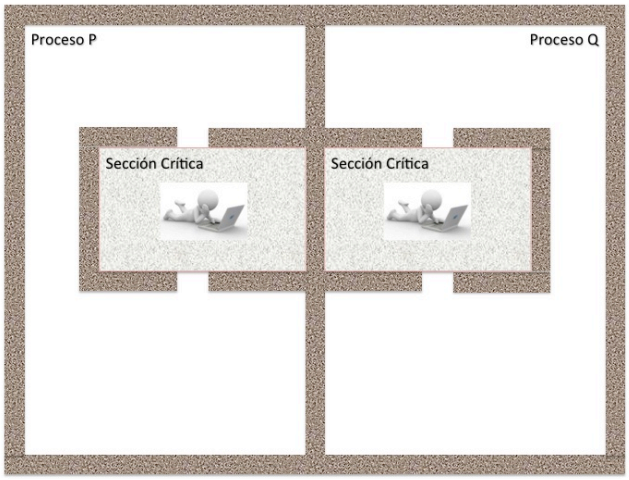

Figura 2. Violación de la exclusión mutua.

Ejercicio 4:

Escriba el código de un programa `p4` que permita simular la Figura 2. El proceso deberá estar en un bucle:

1. *"Caminando por mi habitación"*
2. Al pulsar la tecla `<ENTER>`: *"Intentando entrar en mi Sección Crítica…"*.
3. *"Dentro de mi Sección Crítica"*.
4. Al pulsar la tecla `<ENTER>`: *"He salido de mi Sección Crítica"* (vuelve al punto 1).

Ejecute dos procesos concurrentes de dicho programa y recree un escenario en el que se viole la exclusión mutua.

Primer intento (turno)

Incluya una puerta en cada entrada de la sección crítica, de manera que ambas puertas actúen como un conmutador, si una está abierta la otra está cerrada. Añada un pulsador a la salida de cada sección crítica que hace conmutar las puertas de entrada. Sólo estará activo (color verde) el pulsador de la puerta que está abierta.

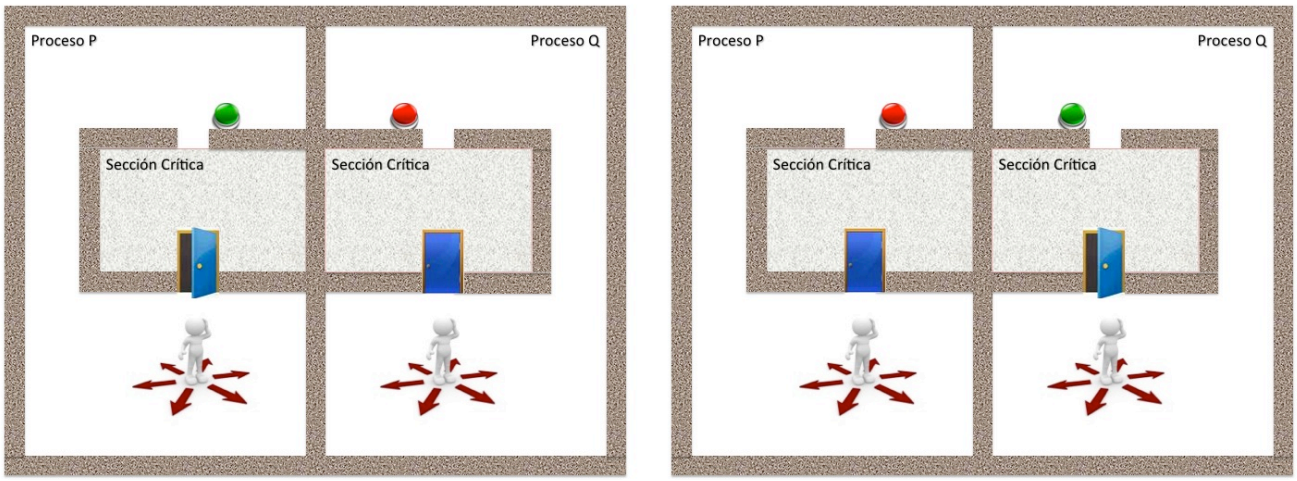

Figura 3. Turno para entrar en la Sección Crítica.

Ejercicio 5:

Escriba un nuevo programa `p5` basado en `p4` que permita simular la Figura 3:

1. *"Caminando por mi habitación"*
2. Al pulsar la tecla `<ENTER>`: *"Intentando entrar en mi Sección Crítica…"*.
   Si la puerta está abierta:
   3. *"Dentro de mi Sección Crítica"*.
   Si la puerta está cerrada:
   3. *"Puerta cerrada"* (vuelve al punto 2).
4. Al pulsar la tecla `<ENTER>`: *"He salido de mi Sección Crítica"*.
5. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"* (vuelve al punto 1).

Ejecute dos procesos concurrentes de dicho programa y compruebe que no es posible violar la exclusión mutua ni producirse interbloqueo.

Recree un escenario en el que uno de los procesos sufra inanición.

Segundo intento (post-solicitud)

Mueva la puerta de entrada a la sección crítica a un pasillo previo. Incluya en dicho pasillo un pulsador que cierra la puerta del otro proceso y que actúa como un conmutador con el pulsador de la salida de la sección crítica, el cual permite abrir la puerta del otro proceso (cuando el pulsador del pasillo está activo (verde), el de la salida de la sección crítica está inactivo (rojo), y viceversa). Inicialmente las dos puertas deberán estar abiertas y los pulsadores de los pasillos activos.

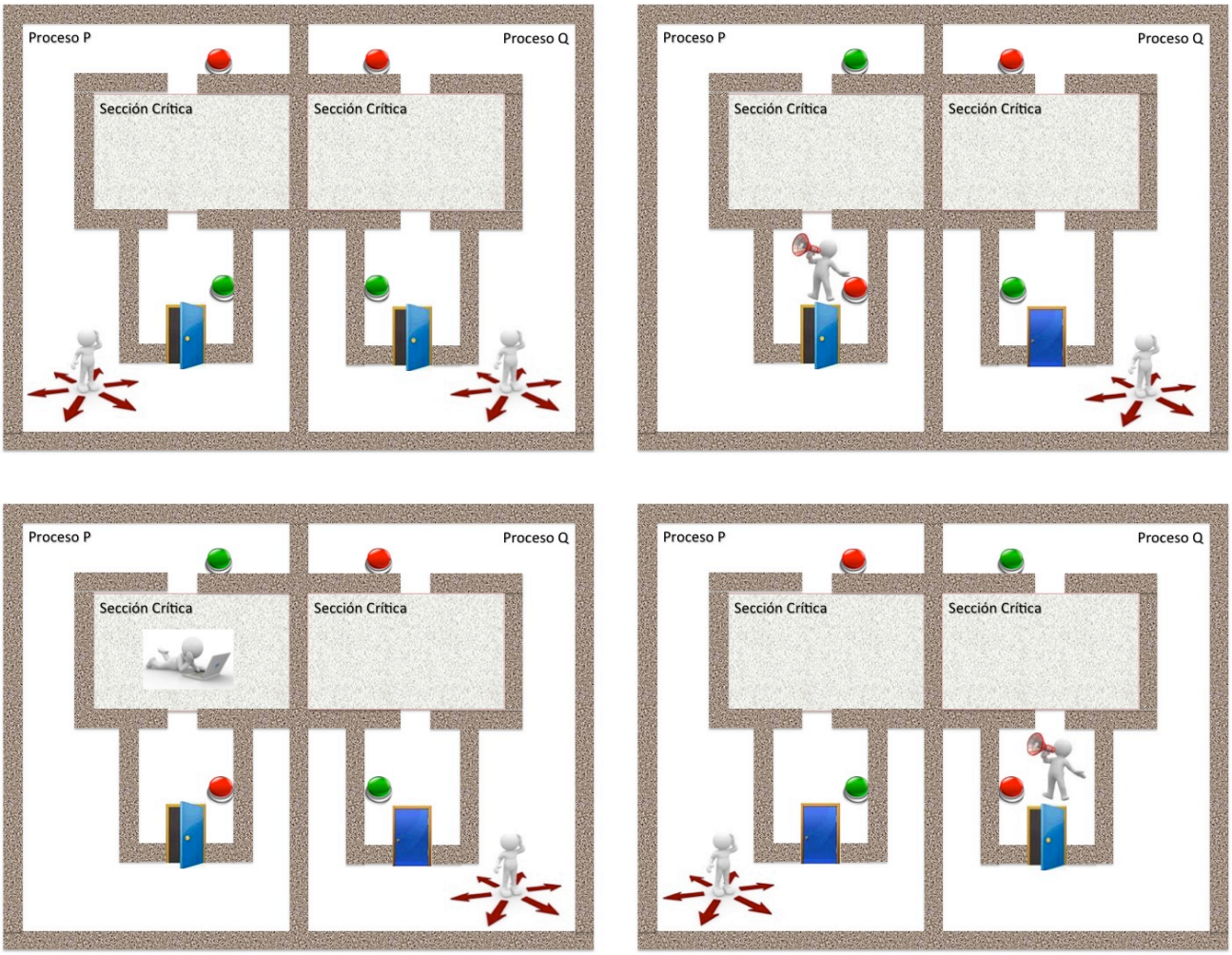

Figura 4. Post-solicitud para entrar en la Sección Crítica.

Ejercicio 6:

Escriba un nuevo programa `p6` basado en `p5` que permita simular la Figura 4:

1. *"Caminando por mi habitación"*
2. Al pulsar la tecla `<ENTER>`: *"Intentando acceder al pasillo de entrada a la Sección Crítica…"*.

   Si la puerta está abierta:

   3. *"Dentro del pasillo"*.

   Si la puerta está cerrada:

   3. *"Puerta cerrada"* (vuelve al punto 2).
4. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"*.
5. *"Dentro de mi Sección Crítica"*.
6. Al pulsar la tecla `<ENTER>`: *"He salido de mi Sección Crítica"*.
7. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"* (vuelve al punto 1).

Ejecute dos procesos concurrentes de dicho programa y compruebe que no es posible que se produzca interbloqueo.

Recree un escenario en el que se viole la exclusión mutua.

Recree un escenario en el que uno de los procesos sufra inanición

Tercer intento (pre-solicitud)

Mueva las puertas de entrada del pasillo a la entrada de la sección crítica, pero mantenga los pasillos con los pulsadores al igual que en el caso anterior. Inicialmente las dos puertas deberán estar abiertas y los pulsadores de los pasillos activos.

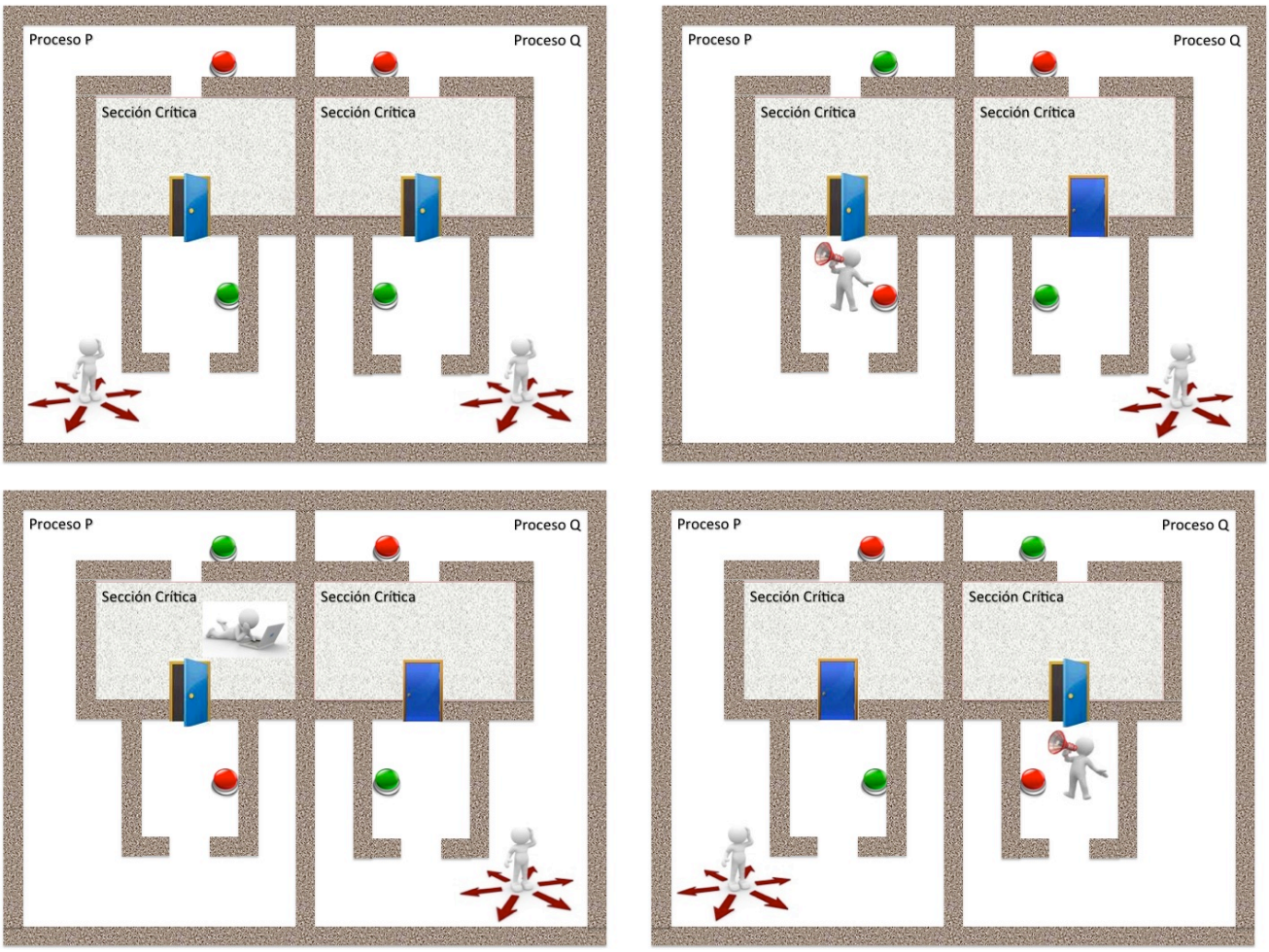

Figura 5. Pre-solicitud para entrar en la Sección Crítica.

Ejercicio 7:

Escriba un nuevo programa `p7` basado en `p6` que permita simular la Figura 5:

1. *"Caminando por mi habitación"*
2. Al pulsar la tecla `<ENTER>`: *"Dentro del pasillo"*.
3. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"*.
4. Al pulsar la tecla `<ENTER>`: *"Intentando acceder a la Sección Crítica…"*.

   Si la puerta está abierta:

   5. *"Dentro de mi Sección Crítica"*.

   Si la puerta está cerrada:

   5. *"Puerta cerrada"* (vuelve al punto 4).
6. Al pulsar la tecla `<ENTER>`: *"He salido de mi Sección Crítica"*.
7. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"* (vuelve al punto 1).

Ejecute dos procesos concurrentes de dicho programa y compruebe que no es posible que se viole la exclusión mutua ni se produzca inanición.

Recree un escenario en el que se produzca una situación de interbloqueo.

Cuarto intento (pre-solicitud + contienda)

Cree una puerta de salida del pasillo que se utilizará en caso de que la puerta a la sección crítica esté cerrada. Dicha puerta se abre y se cierra igual que la puerta de entrada a la sección crítica del otro proceso. Cree un pulsador en la salida del pasillo que funcione y se active igual que el pulsador que está a la salida de la sección crítica. Inicialmente todas las puertas deberán estar abiertas, los pulsadores de entrada a los pasillos activos y el resto de pulsadores inactivos.

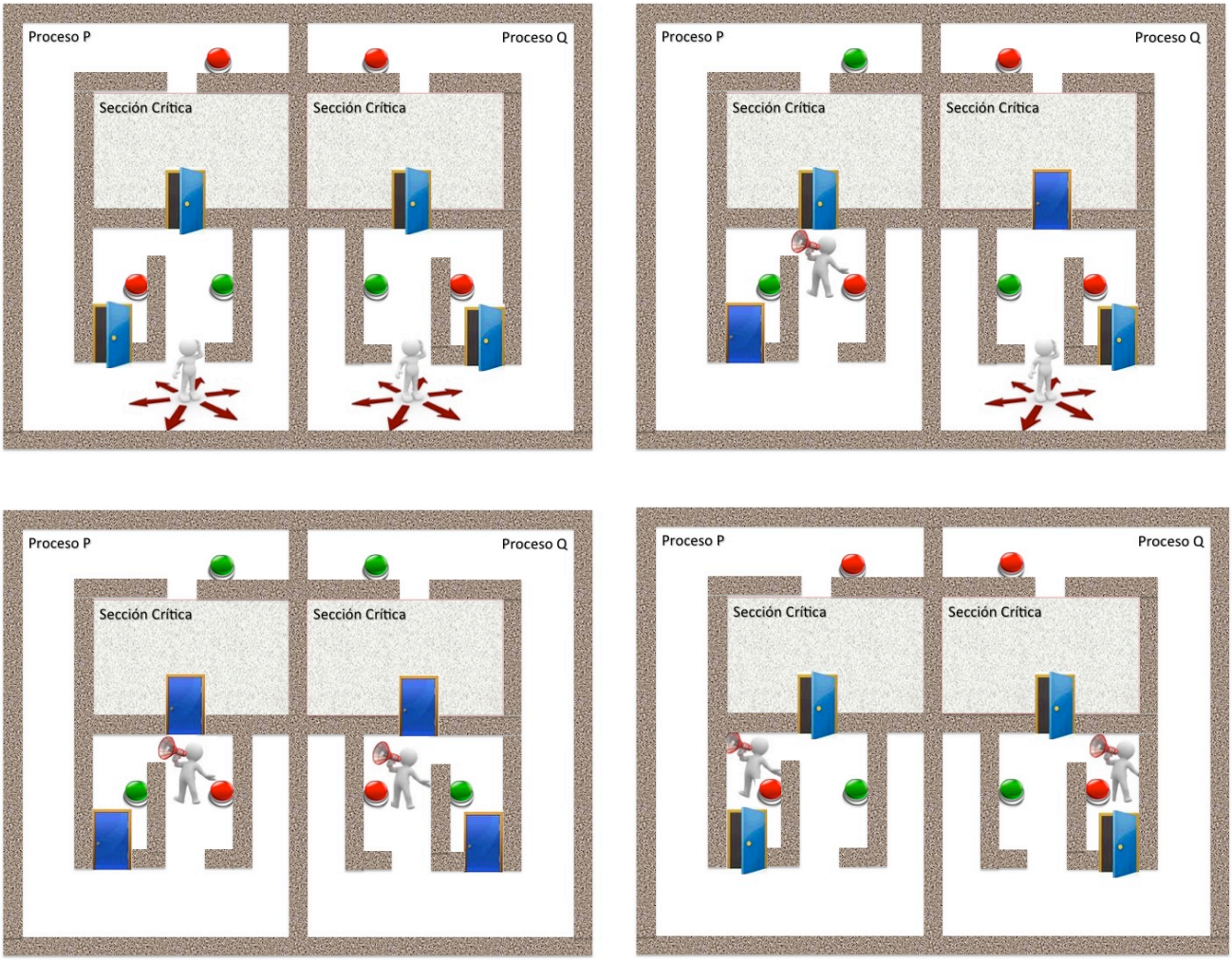

Figura 6. Pre-solicitud + contienda para entrar en la Sección Crítica.

Ejercicio 8:

Escriba un nuevo programa `p8` basado en `p7` que permita simular la Figura 6:

1. *"Caminando por mi habitación"*
2. Al pulsar la tecla `<ENTER>`: *"Dentro del pasillo"*.
3. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"*.
4. Al pulsar la tecla `<ENTER>`: *"Intentando acceder a la Sección Crítica…"*.

Si la puerta está abierta:

5. *"Dentro de mi Sección Crítica"*.

Si la puerta está cerrada:

5.1. *"Puerta cerrada. Saliendo del Pasillo…"*

5.3. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"*

5.4. *"He salido del Pasillo"* (vuelve al punto 1).

6. Al pulsar la tecla `<ENTER>`: *"He salido de mi Sección Crítica"*.
7. Al pulsar la tecla `<ENTER>`: *"He accionado el pulsador"* (vuelve al punto 1).

Ejecute dos procesos concurrentes de dicho programa y compruebe que no es posible que se viole la exclusión mutua ni se produzca interbloqueo.

Recree un escenario en el que se produzca una situación de inanición para uno de los procesos.

Recree un escenario en el que se produzca una situación de inanición para ambos procesos.

Resumen

Los principales resultados esperados de esta práctica son:

- Adquirir experiencia en la programación concurrente de procesos.
- Aprender a utilizar memoria compartida.
- Comprender las problemáticas más comunes en los problemas de concurrencia.

Como trabajo adicional del alumno, se proponen las siguientes líneas:

- Implementar y razonar la solución del algoritmo de Dekker para N procesos.
- Implementar y razonar la solución del algoritmo de Peterson para N procesos.