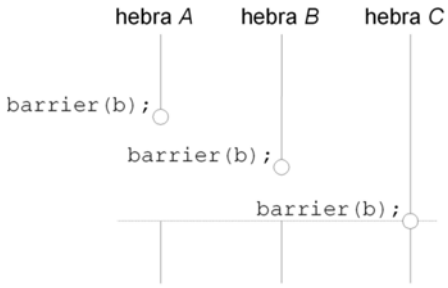
Sistemas en Tiempo Real - P2

Ejercicio 1. Barrera

Vamos a diseñar un objeto de sincronización denominado “barrera”. Las hebras que utilizan una barrera dada, forman un conjunto preestablecido de N elementos, donde N es el parámetro del método de creación de la barrera. En la figura tenemos un ejemplo con N = 3. Todas las hebras van bloqueándose en la barrera, buscando un instante de reanudación común. Éste llega cuando la hebra n-ésima del conjunto invoca la primitiva.



La barrera tiene tres métodos: “*sincronizar*”, “crea” y “destruye”:

- *void crea (barrera_t *b, int N)* → Inicializa las variables mutex y de condición necesarias así como el número de hebras que se bloquean en la barrera (parámetro *N*).
- *void destruye(barrera_t *b)* → Liberará las variables de condición y del mutex.
- *void sincronizar (barrera_t *b)* → Si la hebra invocante no es la n-ésima, se bloquea, si es la n-ésima, libera al resto y continúa la ejecución. En este caso, como la condición de bloqueo solo se debe comprobar una vez, no es necesario usar un bucle *while*, basta con una sentencia *if*.

Se deberá crear un fichero .h con las cabeceras, un .c con el código del paquete de la barrera y otro .c con el programa principal, el cual lanzará 10 hebras que deberán sincronizarse en la barrera. Mostrar mensajes informativos de los sucesos que se crean importantes.

Deberá garantizarse en todo momento la exclusión mutua.

Ejercicio 2. Señales

Implementar un programa que cree un hilo que espere la señal SIGINT. Dicho hilo deberá esperar hasta que el número de ocurrencias de la señal sea 10, momento en el que terminará el hilo y, con él, el programa.

Los ejercicios de la práctica se comprimirán en un único archivo que será entregado a través del campus virtual usando la tarea creada para tal efecto.