PRÁCTICA 3:

PROBLEMA DE LA CENA DE FILÓSOFOS

Sistemas Operativos

Carlos Lozano Alemañy

Rafael Ramírez Berrocal

German Nuñez Sánchez

ÍNDICE

1.INTRODUCCIÓN

Esta práctica trata sobre recrear el famoso problema de la cena de filósofos, propuesto por Dijkstra, en los entornos de Arduino y FreeRtos, ejecutándose en un ESP32. Nuestra solución consiste en no dejar entrar a todos los filósofos a la vez para así evitar el Deadlock (aunque haya una mínima posibilidad de que se produzca por condiciones de carrera). En nuestro programa se diferencia tres partes: la declaración de variables globales y semáforos, la función principal donde se llevará a cabo la actividad de los filósofos y el setup donde se montará todo antes de comenzar la función principal.

2.VARIABLES GLOBALES Y SEMÁFOROS

En este apartado veremos todas las variables que se usarán en todo el programa. En la base del programa se nos otorgó solo el número de filósofos (NUM_OF_PHILOSOPHERS), un tiempo de espera en ms (ESPERA) y el número máximo permitido en la sala (MAX_NUMBER_ALLOWED), siendo al principio restando 1 al número de filósofos totales.

Para comodidad nuestra y para cualquier caso que alguien desee modificarlo, nosotros añadimos uno el cual hará que el programa sea finito o infinito (INFINITO), otro para decidir el número máximo de filósofos permitidos en la sala para forzar o no un Deadlock (DEADLOCK) y un último para decidir si proteger o no los mensajes que se muestren en la pantalla de comandos (MENSAJES).

Otras variables locales creadas fueron un buffer (bufer), el cual indica el número máximo de caracteres imprimibles, el tamaño máximo en Bytes de las tareas (TASK_STACK_SIZE), un array para enumerar a cada uno de los filósofos (filósofos[x], siendo x el valor de NUM_OF_PHILOSOPHERS) y una constante k para las operaciones que se encuentran dentro del setup.

En cuanto a los semáforos, se crean 4 en total: uno encargado de controlar cuantos filósofos entran en la sala (entrarSala), uno para cada uno de los palillos (palillo[x], siendo x el valor de NUM_OF_PHILOSOPHERS), uno para protección de los mensajes (mensajes) y un último encargado de notificar, en caso de ser finito, que todos los filósofos han acabado de comer (done_all).

3.SETUP

Dentro de setup se inicializará y creará todo lo necesario para la función principal. El comienzo de esta es la comunicación del programa con el propio ordenador además de imprimir un mensaje indicando que el programa que mostrará la solución ha iniciado.

Se inicializan los semáforos, creando así las siguientes: dos semáforos Counting, uno para “entrarSala” (que irán quitando o añadiendo valores hasta un máximo de N valor, respecto al número de filósofos que estén permitidos) y otro para “done_all” (que irá añadiendo valores hasta que sea igual al número de filósofos que haya en total), y varios mutex, uno para “mensajes” y los otros para “palillo[x]” (se crearán tantos mutex de “palillo” como número de filósofos haya).

Una vez los semáforos están listos, se crean los N filósofos existentes y entran a la función principal para comenzar su actividad.

Como añadido para el caso finito, hay un controlador, usando el semáforo “done_all” que esperará a que todos hayan acabado de comer, mostrando después un mensaje anunciando el final del programa.

4.FUNCIÓN PRINCIPAL: “Filosofo”

Al comienzo de esta función, y antes de comenzar toda actividad del filósofo, se guarda el número asignado al filósofo correspondiente (numero), se le asigna sus dos palillos correspondientes (idx_1 para el izquierdo y idx_2 para el derecho) y se crea un randomSeed para crear números aleatorios entre 0 y el valor de ESPERA.

Para la actividad de los filósofos se crea un “do while”, que hará un bucle infinito en caso de que el valor de INFINITO sea igual a 1, en caso contrario acabará toda la actividad y dará un aviso de su finalización en el semáforo “done_all”. Dentro del bucle encontramos 10 acciones distintas que llevarán a cabo los filósofos:

Primera acción: El filósofo se queda T tiempo pensando.

Segunda acción: El filósofo pide entrar a comer.

Tercera acción: El filósofo tarda T tiempo en entrar a comer, más si toda la sala está llena (controlada por “entrarSala”, decrementando su valor).

Cuarta acción: El filósofo tarda T tiempo en tomar el palillo izquierdo, más si este ya ha sido tomado por otro filósofo (controlado por “palillo[X]”, siendo X el valor del idx_1 del filósofo N).

Quinta acción: El filósofo tarda T tiempo en tomar el palillo derecho, más si este ya ha sido tomado por otro filósofo (controlado por “palillo[X]”, siendo X el valor del idx_2 del filósofo N).

Sexta acción: El filósofo tarda T tiempo en comenzar a comer.

Séptima acción: El filósofo tarda T tiempo para acabar de comer.

Octava acción: El filósofo tarda T tiempo en dejar el palillo izquierdo (dejando “palillo[X]” libre).

Novena acción: El filósofo tarda T tiempo en dejar el palillo derecho (dejando “palillo[X]” libre).

Décima acción: El filósofo tarda T tiempo en salir de la sala (incrementando después el valor de “entrarSala”).

Cabe destacar que, para cada acción mencionada anteriormente, se imprime un mensaje de que el filósofo está llevando a cabo o ha realizado dicha acción, siendo estos protegidos o no (decidido por medio de MENSAJES) por medio del semáforo “mensajes” para evitar su superposición.

5.FUNCIONAMIENTO

En este apartado veremos imágenes de nuestro programa ejecutándose en el ESP32. Se han realizado varias pruebas cambiando el número de filósofos y si se llevan a cabo o no un Deadlock, la protección de los mensajes y una ejecución finita, no obstante, no se mostrarán todas las pruebas realizadas, solo unas cuantas de estas:

```
--- SOLUCIÓN FreeRTOS COMIDA DE FILÓSOFOS---
Filósofo 0 se ha creado
Filósofo 1 se ha creado
Filósofo 2 se ha creado
Filósofo 3 se ha creado
Filósofo 0 está pensando
Filósofo 4 se ha creado
Filósofo 3 está pensando
Filósofo 1 está pensando
Filósofo 2 está pensando
Filósofo 4 está pensando
Filósofo 2 quiere entrar a comer
Filósofo 0 quiere entrar a comer
Filósofo 1 quiere entrar a comer
Filósofo 3 quiere entrar a comer
Filósofo 0 se ha sentado a comer
Filósofo 4 quiere entrar a comer
Filósofo 2 se ha sentado a comer
Filósofo 2 coge palillo izquierdo 2
Filósofo 0 coge palillo izquierdo 0
Filósofo 1 se ha sentado a comer
Filósofo 4 se ha sentado a comer
Filósofo 4 coge palillo izquierdo 4
Filósofo 1 coge palillo izquierdo 1
Filósofo 2 coge palillo derecho 3
Filósofo 2 está comiendo
Filósofo 2 ha acabado de comer
Filósofo 2 deja palillo izquierdo 2
Filósofo 1 coge palillo derecho 2
Filósofo 1 está comiendo
Filósofo 2 deja palillo derecho 3
Filósofo 2 se va de la sala
Filósofo 3 se ha sentado a comer
Filósofo 1 ha acabado de comer
Filósofo 1 deja palillo izquierdo 1
Filósofo 0 coge palillo derecho 1
Filósofo 1 deja palillo derecho 2
Filósofo 1 se va de la sala
Filósofo 0 está comiendo
Filósofo 3 coge palillo izquierdo 3
Filósofo 0 ha acabado de comer
Filósofo 0 deja palillo izquierdo 0
Filósofo 4 coge palillo derecho 0
Filósofo 0 deja palillo derecho 1
Filósofo 4 está comiendo
Filósofo 0 se va de la sala
Filósofo 4 ha acabado de comer
Filósofo 4 deja palillo izquierdo 4
Filósofo 3 coge palillo derecho 4
Filósofo 3 está comiendo
Filósofo 4 deja palillo derecho 0
Filósofo 3 ha acabado de comer
Filósofo 4 se va de la sala
Filósofo 3 deja palillo izquierdo 3
Filósofo 3 deja palillo derecho 4
Filósofo 3 se va de la sala
Todos los filósofos han comido, NO DEADLOCK!
```

· En esta imagen podemos ver la ejecución finita del programa con un total de 5 filósofos, sin que se produzca un Deadlock y con los mensajes están protegidos para evitar una superposición.

```
Filósofo 2 ha acabado de comer
Filósofo 3 se va de la sala
Filósofo 3 está pensando
Filósofo 2 deja palillo izquierdo 2
Filósofo 1 coge palillo derecho 2
Filósofo 2 deja palillo derecho 3
Filósofo 4 quiere entrar a comer
Filósofo 3 quiere entrar a comer
Filósofo 3 se ha sentado a comer
Filósofo 3 coge palillo izquierdo 3
Filósofo 4 se ha sentado a comer
Filósofo 3 coge palillo derecho 4
Filósofo 1 está comiendo
Filósofo 2 se va de la sala
Filósofo 2 está pensando
Filósofo 2 quiere entrar a comer
Filósofo 2 se ha sentado a comer
Filósofo 1 ha acabado de comer
Filósofo 3 está comiendo
Filósofo 3 ha acabado de comer
Filósofo 1 deja palillo izquierdo 1
Filósofo 0 coge palillo derecho 1
Filósofo 1 deja palillo derecho 2
Filósofo 0 está comiendo
Filósofo 3 deja palillo izquierdo 3
Filósofo 1 se va de la sala
Filósofo 1 está pensando
Filósofo 3 deja palillo derecho 4
Filósofo 4 coge palillo izquierdo 4
Filósofo 0 ha acabado de comer
Filósofo 2 coge palillo izquierdo 2
Filósofo 2 coge palillo derecho 3
Filósofo 0 deja palillo izquierdo 0
Filósofo 0 deja palillo derecho 1
Filósofo 2 está comiendo
Filósofo 1 quiere entrar a comer
```

· En esta imagen podemos ver un fragmento la ejecución infinita del programa. Al estar continuamente funcionando, es imposible mostrar toda la actividad de esta.

```
Filósofo 2 ha acabado de comer
Filósofo 2 deja palillo izquierdo 2
Filósofo 0 deja palillo derecho 1
Filósofo 1 coge palillo izquierdo 1
Filósofo 3 se ha sentado a comer
Filósofo 2 deja palillo derecho 3
Filósofo 1 coge palillo derecho 2
Filósofo 2 se va de la sala
Filósofo 2 está pensando
Filósofo 0 se va de la sala
Filósofo 0 está pensando
Filósofo 3 coge palillo izquierdo 3
Filósofo 1 está comiendo
Filósofo 0 quiere entrar a comer
Filósofo 2 quiere entrar a comer
Filósofo 3 coge palillo derecho 4
Filósofo 1 ha acabado de comer
Filósofo 2 se ha sentado a comer
Filósofo 3 está comiendo
Filósofo 1 deja palillo izquierdo 1
Filósofo 1 deja palillo derecho 2
Filósofo 2 coge palillo izquierdo 2
Filósofo 1 se va de la sala
Filósofo 0 se ha sentado a comer
Filósofo 1 está pensando
Filósofo 3 ha acabado de comer
Filósofo 3 deja palillo izquierdo 3
Filósofo 2 coge palillo derecho 3
Filósofo 3 deja palillo derecho 4
Filósofo 1 quiere entrar a comer
Filósofo 0 coge palillo izquierdo 0
Filósofo 2 está comiendo
Filósofo 2 ha acabado de comer
Filósofo 0 coge palillo derecho 1
Filósofo 1 se ha sentado a comer
Filósofo 3 se va de la sala
Filósofo 3 está pensando
Filósofo 2 deja palillo izquierdo 2
Filósofo 0 está comiendo
Filósofo 2 deja palillo derecho 3
Filósofo 0 ha acabado de comer
Filósofo 3 quiere entrar a comer
Filósofo 0 deja palillo izquierdo 0
Filósofo 0 deja palillo derecho 1
Filósofo 1 coge palillo izquierdo 1
Filósofo 2 se va de la sala
Filósofo 2 está pensando
Filósofo 0 se va de la sala
Filósofo 0 está pensando
Filósofo 2 quiere entrar a comer
Filósofo 0 quiere entrar a comer
Filósofo 3 se ha sentado a comer
Filósofo 0 se ha sentado a comer
Filósofo 2 se ha sentado a comer
Filósofo 2 coge palillo izquierdo 2
Filósofo 3 coge palillo izquierdo 3
Filósofo 0 coge palillo izquierdo 0
```

· En esta imagen se muestra un fragmento de la ejecución infinita del programa detenida por un Deadlock. Al estar continuamente funcionando, es imposible mostrar toda la actividad de esta, por lo que en este caso se muestra el final.

```
--- SOLUCIÓN FreeRTOS COMIDA DE FILÓSOFOS---
Filósofo 0 se ha creado
Filósofo 1 se ha creado
Filósofo 2 se ha creado
Filósofo 0 está pensando
Filósofo 2 está pensando
Filósofo 1 está pensando
Filósofo 0 quiere entrar a comer
Filósofo 2 quiere entrar a comer
Filósofo 2 se ha sentado a comer
Filósofo 1 quiere entrar a comer
Filósofo 1 se ha sentado a comer
Filósofo 0 se ha sentado a comer
Filósofo 0 coge palillo izquierdo 0
Filósofo 2 coge palillo izquierdo 2
Filósofo 1 coge palillo izquierdo 1
```

· En esta imagen se muestra la ejecución finita del programa detenida por un Deadlock. A pesar de que fuera finita, se llega a notar que cuantos menos filósofos haya, más fácil es que se produzca el Deadlock.

```
--- SOLUCIÓN FreeRTOS COMIDA DE FILÓSOFOS---
Filósofo 0 se ha creado
Filósofo 1 se ha creado
Filósofo 2 se ha creado
Filósofo 1 está pensanFilósofo 1 está pensando
Filósofo 1 está pensando
Filósofo 1 está pensando

Filósofo 3 está pentrar Filósofo 1 quiere entra
Filósofo 1 quiere entrar a comer

Filósofo 5 se ha creado
Filósofo 6 se ha creado
Filósofo 0 quiere entrar Filósofo 0 quiere entrar Filósofo 6 está pensando comer
Filósofo 6 está pensando comer
Filósofo 1 se ha sentado Filósofo 1 se ha sentado a comer



Filósofo 1 se ha sentado a comer
Filósofo 3 se ha sentado a comer
Filósofo 5 quiere entado a comerFilósofo 2 se ha sentado a comer
Filósofo 2 se ha sentado a comer 1
Filósofo 2 se ha sentado a comer
Filósofo 5 se ha sentado a comer

Filósofo 5 se ha sentado a comerFilósofo 5 se ha sentado a comer 3Filósofo 5 se ha sentado a comerFilósofo 6 se ha sentado a comer



Filósofo 6 se ha sentado a comer 2
Filósofo 6 se ha senllo izquierdFilósofo 5 coge palillo izquierdo 5
Filósofo 5 coge palillo izquierdo 5

Filósofo 6 coge palillo izquierdo 6
```

·En sesta imagen es la ejecución del programa con un total de 7 filósofos. Se muestra la superposición que se producen en los mensajes al no estar protegidos por el semáforo “mensajes”.

6.CONCLUSIÓN

La realización de este trabajo, a pesar de su dificultad, ha mostrado ser bastante entretenida y agradable. A demás, al ser una actividad nueva dentro de la asignatura de Sistemas Operativos, es una gran oportunidad de comenzar la programación de microcontroladores tales como el utilizado ahora, el ESP32.