Sistemas Distribuídos

Práctica Final: Problema de Santa Claus (ampliado)

Descripción del problema.
En esta práctica final ampliamos el funcionamiento de la práctica 2. Ahora los elfos se dividen en dos batallones de 16 elfos, de los cuales uno se comporta como líder. Cada líder tiene una caché de regalos que replica la información de un almacén principal. La función del líder es obtener los regalos del almacén y entregárselos a los demás elfos de su batallón, y después actualizar las cachés siguiendo un protocolo *write-through*.

Las condiciones de finalización ahora son o bien que lleguen todos los renos o bien que se hayan fabricado todos los regalos.

~

Estrategias y mecanismos de sincronización.
Almacén y cachés:
Los líderes tienen que tener acceso a todas las cachés y al almacén, por lo que he optado por crear las estructuras e inicializarlas en el *main()* antes de crear las gorrutinas de los líderes y pasárselas por referencia. Cada líder sabe cuál es su caché porque también le pasan la variable sobre la que itera el bucle que los lanza, y dentro del *array* de cachés le corresponde esa posición.

Con respecto al almacén, los valores booleanos de las cachés están invertidos ya que en el almacén representan si hay un regalo en esa posición, mientras que en las cachés representan si está disponible esa posición (no hay ni ha habido regalo ahí todavía).

Santa:
A diferencia de la práctica 2, he utilizado canales para despertar a Santa. Los renos lo despiertan por un canal y los elfos por otros. Además hay un tercer canal para que los líderes avisen de que ya han terminado con todos los regalos. Utilizando la estructura select y tres canales queda algo más organizado y mejora la legibilidad del código.

Cuando un elfo tiene problemas con un regalo, lo deja hasta que Santa lo arregle (*time.Sleep*) y mientras tanto continúa con otros regalos. Es decir, la gorrutina no se queda esperando a Santa (a excepción del tercer elfo el cual necesariamente espera a Santa para acto seguido actualizar la cuenta de elfos en problemas y de regalos totales terminados).

Una vez han llegado todos los renos, Santa, antes de irse con ellos, deja una nota en la puerta de su dormitorio para que los elfos la lean (variable compartida, un *flag*). La nota es necesaria para evitar que los elfos se queden bloqueados intentando

despertar a Santa, ya que nunca responderá. Los líderes, de camino al almacén, pasan por delante del dormitorio de Santa, y si ven la nota avisarán a todos los elfos para que paren de trabajar, ya que los regalos que fabriquen no serían entregados este año y tendrían pérdidas en *stock*. Los elfos cuando vayan a pedirle ayuda a Santa leerán la nota y se irán.

Leader:
Como expliqué antes, recibe el *array* de cachés y su "índice" de líder para saber cuál de las cachés es la suya. También recibe el almacén, un canal creado en en la función que lanza los batallones para comunicarse con los elfos de su batallón, y el canal para notificar a Santa cuando hayan terminado de fabricar regalos.

La función entra en un bucle del que saldrá cuando haya terminado de coger todos los regalos del almacén, o cuando Santa se haya ido y el líder lea la nota. A continuación, entra en otro bucle que modela la secuencia de selección de filas del almacén, y en una variable se anotan las filas ya seleccionadas, reseteándola cada vez que se completa el ciclo.

Tras elegir una fila, se elige al azar uno de los treinta regalos. Accede a su caché para comprobar su disponibilidad. Si está disponible, lo coge. Si no está disponible, en vez de elegir otro al azar, prueba con el de índice siguiente. De esta manera se evitan retrasos potencialmente largos en la selección de un regalo, aunque algunas posiciones tendrán más posibilidades que otras de ser elegidas.

A continuación actualiza las cachés y el almacén, y envía el regalo a los elfos por su canal correspondiente. Tras la entrega de cada regalo la función duerme un par de segundos para poder hacer un buen seguimiento de la ejecución del programa.

Una vez se han entregado todos los regalos o Santa se ha ido, informa a los elfos de su batallón para que dejen de trabajar. En el primer caso además informará a Santa y a la función que crea los renos para que terminen también.

Elfos:
A diferencia de la práctica 2, ahora los elfos fabricarán regalos en bucle hasta que su líder los avise. Reciben los regalos del líder por un canal, y si todo va bien aumentan la variable de regalos totales fabricados. En caso de necesitar ayuda, aumentarán la variable del número de elfos en problemas, o si es el tercer elfo que necesita ayuda, comprobará si está Santa (leyendo la nota) y en caso de estar disponible lo despertará. El tercer elfo esperará a que Santa termine con los tres regalos y los añadirá de golpe a la variable de regalos totales.

Renos:
Si no es el último reno, simplemente aumentará la variable de renos que ya han llegado. Si es el último, tras aumentarla, irá a despertar a Santa. El reno esperará

hasta que Santa haya terminado de escribir la nota y mandar sus mensajes para soltar el *lock*, ya que de otra manera podría colarse un elfo para despertar a Santa antes de poder ver que Santa se ha ido.

SendoffElves:
Esta función la ejecuta el main en paralelo para crear los batallones de elfos, incluyendo los líderes. Recibe el almacén y el *array* de cachés y se los pasa a los líderes, indicándoles cuál es su caché correspondiente.

SendoffRdeers:
Similar a su versión en la práctica 2 a excepción de que ahora antes de lanzar un nuevo reno comprueba si se han terminado de fabricar los regalos, en cuyo caso muere sin lanzarlo.

Wait group:
Al terminar primero los elfos, el programa tarda unos segundos en acabar ya que se espera por las *gorrutinas* activas, entre ellas *sendoffRdeers*, la cual está en un *time.Sleep* y hasta que no pasen unos segundos no podrá enterarse de que debe terminar su ejecución. Este retardo creo que es inevitable si queremos asegurarnos de que las *gorrutinas* acaban de forma ordenada.