Miguel Ángel Sánchez
Juan Velasco

miquelangel.sanchezm@estudiante.uam.es
juan.velascoi@estudiante.uam.es

Proyecto final: Memoria

Date: 08/05/18

Vamos a ir parte por parte comentando nuestras decisiones de diseño y sobre todo la forma en la que se comunican los distintos procesos.

Simulación de la carrera

El proceso principal (ya dentro de la función carrera) crea tantos procesos caballo como caballos haya.

Este proceso manda a cada caballo, a través de una tubería, el tipo de tirada que tiene que hacer. Los caballos tiran el dado teniendo en cuenta el tipo y le devuelven el resultado al proceso principal a través de un mensaje a una cola de mensajes. Por su parte, el proceso principal lee estas tiradas y actualiza la posición de los caballos en la memoria compartida que utiliza junto al proceso monitor (memoria que está protegida con el semáforo 0).

Esto se repite en bucle hasta que un caballo traspase la meta, momento en el que terminará la carrera, y el caballo que más adelantado esté será el ganador. En caso de que haya dos caballos como posibles ganadores, nos hemos quedado con el de id más bajo.

En el pdf se pedía implementar que la carrera pudiese terminarse también pulsando control c. Viendo que la carrera se realiza a una velocidad muy elevada (incluso poniendo una meta bastante alejada) y que es prácticamente imposible mandar ninguna señal durante la carrera, hemos decidido no implementar esta petición.

Monitor

Lo hemos dividido en tres funciones:

monitor_antes: Es el que lleva la cuenta atrás para que comience la carrera y el encargado a su vez de mostrar por pantalla como van las cotizaciones (Que, al ser memoria compartida con el proceso gestor de apuestas, están protegidas por el semáforo 1)

Cuando acaba la cuenta atrás, el monitor manda una señal al padre y lo despierta del pause en el que está para que comience la carrera.

monitor_durante: Es el que va imprimiendo la posición de los caballos (accediendo a ella no sin antes bajar el semáforo 0 y subiéndolo después) y su última tirada (que está en la misma memoria compartida). Cuando un caballo pasa la línea de meta se pasa al monitor_despues.

monitor_despues: Comienza con una cuenta atrás de los 15 segundos que se piden que haya antes de mostrar los resultados de la carrera.

Luego muestra los resultados de la carrera, con la posición de cada caballo, y la lista de como mucho 10 apostadores con más beneficios.

El tema de las apuestas es información que se tiene en una memoria compartida con el proceso Gestor de apuestas. En monitor_antes está protegida con el semáforo 1, pero como en este momento de la ejecución estamos seguros de que ya ningún otro proceso va a acceder a esa memoria (pues el único proceso que podía hacerlo, el gestor, está ya muerto), hemos decidido no poner el semáforo.

Esta función además imprime en un fichero (protegido por el semáforo 2) el report. Hace falta proteger el fichero porque también el proceso gestor de apuestas escribirá en él y sus ventanillas lo harán concurrentemente.

El monitor, antes de terminar, le manda una señal al proceso padre para decirle que ya puede eliminar los recursos porque nadie más los va a usar.

Apostador

Este es un proceso muy simple que cada segundo crea una apuesta de cantidad y caballo aleatorios siendo siempre la cantidad menor que max_dinero y el caballo menor que num_caballos.

Se generan de forma secuencial, es decir, los apostadores van llegando por orden y cada uno solo realiza una apuesta. Solo se pueden hacer por tanto tantas apuestas como apostadores haya, o 30 si el número de apostadores es mayor que 30, ya que solo se puede generar una apuesta por segundo y la cuenta atrás es de 30 segundos.

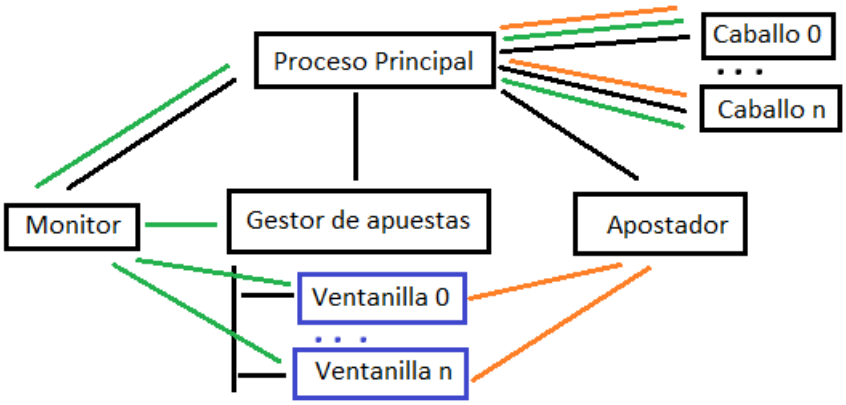
Gestor de apuestas

El gestor de apuestas inicializa las apuestas y crea ventanillas (threads) que van leyendo apuestas de la cola de mensajes y procesándolas.

Es importante comentar el uso del semáforo 4 en este proceso. Este semáforo sirve para que cada ventanilla pueda saber qué número de ventanilla tiene y pueda, por tanto, imprimirlo en el report.

Tanto el proceso apostador como el gestor de apuestas son eliminados por el proceso padre cuando va a empezar la carrera mandándoles una señal.

Pequeño esquema



En negro se muestra la creación de procesos. Como vemos el proceso principal crea todos los procesos. Luego el gestor crea los hilos ventanilla que están en azul.

Los procesos que están unidos por una flecha verde se comunican por memoria compartida, mientras que los que están unidos por una flecha naranja se comunican por una cola de mensajes.

Hay que aclarar que la cola de mensajes entre ventanillas y apostador es diferente a la cola de mensajes entre caballos y proceso principal. Al igual que la memoria compartida entre ventanillas, gestor y monitor es diferente a la memoria compartida entre monitor, proceso principal y caballos.