SISTEMAS OPERATIVOS: Informe práctica 4

Ejercicio 1: cola de mensajes

En este ejercicio se pedía crear tres procesos en el que el primero lee el texto de un fichero, lo trocea y envía cada trozo en un mensaje. El segundo proceso recibe los mensajes, lo convierte a mayúsculas y envía el texto a la cola de mensajes. Finalmente, el tercer proceso recibe estos mensajes y los escribe en un fichero. El nombre de los ficheros se debe pasar como argumentos de entrada, aunque por defecto, el fichero de salida se denomina "destino.txt".

En nuestra implementación, los procesos se crean secuencialmente. Primero se crea el proceso que lee del fichero y trocea el texto. Para trocearlo, hemos decidido partir el texto en partes iguales (salvo el último que probablemente sea más corto), teniendo en cuenta la longitud del texto (usando las funciones fseek y ftell) y el tamaño máximo de bytes que puede almacenar la cola de mensajes, almacenado en la estructura msqid_ds. De esta forma, nos aseguramos de que no se sobrepase el límite, dado que cada mensaje ocupa 4KB. Una vez se han mandado todos los mensajes, se crea el siguiente proceso que los lee en un bucle y cada texto lo convierte a mayúsculas. Para conseguir este paso, simplemente se han usado los caracteres ASCII, dado que la distancia entre cada letra y su correspondiente letra mayúscula es constante. Finalmente, después de que se hayan enviado estos mensajes, se crea el tercer proceso que los lee y los escribe en el fichero de texto pasado en la línea de comandos. Si solo se ha pasado el fichero origen, se creará un fichero "destino.txt" donde se escribirá el resultado final.

Ejercicio 2: carrera de caballos

En este programa, se pedía simular una carrera de caballos con un gestor de apuestas, apostadores y ventanillas, un monitor y los caballos. Para realizarlo, el programa principal crea un proceso, que será el gestor de apuestas, y un hilo que será el monitor y espera quince segundos. En este tiempo, el hilo imprime los segundos que han pasado y se generan las apuestas. El gestor, crea la cola de mensajes, el array de semáforos y tantos hilos como ventanillas, las cuales se encargarán de recibir las apuestas y actualizar las cotizaciones de cada caballo. Posteriormente, crea un proceso que genera las apuestas de forma aleatoria y espera a que el proceso padre le envíe una señal (SIGUSR2 que actualiza el estado de la carrera) para cerrar las ventanillas y las apuestas, liberar todos los recursos y terminar el proceso gestor. Las apuestas se almacenan en una estructura que contiene el identificador del mensaje, el nombre del apostador, el número del caballo al que se apuesta y la cantidad apostada. Estos tres últimos datos se generan de forma aleatoria mediante la función rand(), se envía a la cola de mensajes y espera 0.1 segundos. Simultáneamente, cada hilo ventanilla recibe una estructura que contiene el número de caballos y el de apostadores, para comprobar que la apuesta es válida, una matriz en la que se almacena la cantidad que apuesta cada apostador a cada caballo, para ver al final cuánto gana cada apostador, el estado de la carrera (-1 si no ha empezado o EMPEZADA), el total apostado, el id de la cola de mensajes y el de los

semáforos y un array de estructuras que contienen el id, total apostado y cotización de un caballo. Esta estructura está en memoria compartida, de manera que el hilo monitor también tenga acceso a él y pueda imprimir la cotización de cada caballo justo antes de empezar la carrera y los beneficios y apuestas ganadoras al final de la misma. Hay tantos semáforos como caballos y número de apostadores más uno, todos ellos binarios. El primero sirve para que solo una ventanilla pueda leer y modificar el total apostado a todos los caballos. Los siguientes sirven para que únicamente una ventanilla pueda acceder y modificar los datos del caballo correspondiente a la apuesta recibida. Y los últimos sirven para que solo una ventanilla pueda cambiar los datos del apostador correspondiente a la apuesta recibida. De esta manera, puede haber varias ventanillas trabajando a la vez, siempre y cuando no quieran acceder al mismo caballo o al mismo apostador. Otra solución habría sido utilizar un único semáforo para controlar el acceso a la memoria compartida, pero entonces solo podría haber una ventanilla trabajando en cada instante y reduciría considerablemente la eficiencia. Cada ventanilla se encarga de recoger un mensaje de la cola, hacer un down del semáforo correspondiente a ese caballo y otro down correspondiente al semáforo de ese apostador. Posteriormente, se actualiza la matriz para sumar a la casilla que relaciona el apostador con el caballo la cantidad apostada. Se hace un down del primer semáforo para acceder al dato del total apostado a todos los caballos para sumarle la cantidad apostada. Se hace el up del primer semáforo y del semáforo del apostador. Antes de hacer el up del semáforo del caballo, se actualiza el dato total apostado y su cotización. Todo ello se realiza en un bucle hasta que el estado pasa a EMPEZADO, en cuyo caso, sale del bucle y termina de ejecutarse el hilo y el proceso gestor de apuesta.

En este instante, el hilo monitor imprime las cotizaciones de cada caballo y empieza la carrera. El proceso principal crea tantos hijos como caballos inicializando sus posiciones y tiradas a cero y entran en un bucle infinito y se bloquean hasta recibir la señal SIGUSR1 del padre. Mientras, el padre escribe en la tubería la posición del primer caballo, le manda la señal al hijo y se bloquea a la espera de que el hijo le devuelva la señal. En este momento, el hijo lee de la tirada su posición y la posición del último caballo, realiza la tirada en función de estos datos y escribe el número resultante en otra tubería y se vuelve a bloquear después de enviarle la señal al padre. El padre lee de la otra tubería y actualiza la tirada del caballo. Así sucesivamente con el resto de caballos hasta que todos hayan tirado. Después, el padre ordena las posiciones de los caballos y espera a que el monitor escriba por pantalla las posiciones de los caballos después de cada tirada. En caso de que alguno de los caballos haya superado la longitud de la carrera o que se haya recibido la señal SIGINT en algún instante entre medias de las tiradas, la carrera se dará por FINALIZADA. En este caso, el monitor imprime el podio de los caballos ganadores, las apuestas ganadoras y los beneficios acumulados. Finalmente, el padre mata todos los procesos hijo y libera todos los recursos. Al comprobar el funcionamiento del programa, hemos observado que, en ciertas ocasiones, el proceso padre se quedaba bloqueado. Para solucionarlo, el proceso monitor comprueba si al cabo de cinco segundos, el proceso monitor ha imprimido algo. En caso negativo, le envía la señal

SIGUSR1 al proceso principal. Por este motivo, en alguna ocasión, tarda más de lo normal en imprimir el hilo monitor.

Durante la realización de este programa tuvimos que tomar diversas decisiones. La estructura del hilo de las ventanillas se decidió escribir en memoria compartida, para que tanto el gestor de apuestas como el hilo monitor tuviesen acceso a esos datos. Además, el monitor se decidió hacer como un hilo y no como un proceso para que tuviese más fácil acceso al cambio de posiciones de los caballos y pudiese imprimirlo correctamente. A su vez, el monitor imprime la cotización de los caballos una vez ha finalizado el tiempo de las apuestas, porque nos parecía más lógico para saber qué caballo es el favorito. Finalmente, decidimos que el monitor imprimiese únicamente la posición de los caballos después de cada tirada, dado que, en otro caso, no serían muy fiables las posiciones y no se sabría cuáles tienen una tirada más. En cuanto a la modularización del programa, hemos realizado una función para el gestor de apuestas, para el apostador, para cada ventanilla, para el monitor y una función dado que ejecuta cada hijo y devuelve la tirada. De esta manera, es más sencillo realizar modificaciones o ampliaciones en el programa.