Grupo : 2202
Asignatura : Sistemas operativos
Alumnos : Lucía Asencio y Rodrigo De Pool

Introducción:

En el siguiente documento se encuentra un breve resumen de las decisiones de implementación tomadas a lo largo de la práctica.

Ejercicio 1:

Se puede comprobar el correcto funcionamiento del programa utilizando como prueba el fichero ‘f1’ dado y utilizando el programa ‘valgrind’. Podemos ver que la ejecución está libre de fugas de memoria.

Ejercicio 2:

En este ejercicio, hemos tenido catorce mil millones de problemas para conseguir que Valgrind nos dejara en paz, pero al final hemos conseguido una ejecución correcta del programa.

En primer lugar, al ejecutar con valgrind teníamos warnings en todos nuestros mensajes que nos avisaban de un posible error por trabajar con mensajes no inicializados (aunque todas las estructuras de mensajes eran tarde o temprano inicializadas). Por esto, decidimos inicializar las estructuras a {0}, antes de darles el valor deseado, para evitar los warnings. El funcionamiento de los mensajes es correcto.

Con estas inicialaciones, la única pega que nos ha puesto valgrind es con respecto a memoria proveniente de la cancelación de los threads. Hemos probado el uso de manejadores en threads (sin éxito). Hemos conseguido que gran parte de estas pérdidas desaparecieran con el uso de pthread_detach antes de pthread_cancel, pero ninguna otra medida sirvió de nada.

Hemos comprobado el uso de colas con el comando “ipcs” sobre la terminal, y nuestras colas son correctamente liberadas al completarse la ejecucion del programa.

En las salidas aparece que todos los participantes tienen ganancias en la carrera, y esto tiene una explicación: lo que muestra por pantalla es simplemente la suma de las ganancias de cada apostador con los 3 caballos ganadores, sin restar la suma de todo el dinero apostado por ese apostador – es por esto que los valores son estrictamente positivos. Como en el tiempo de espera anterior a la carrera se suceden muchisimas apuestas, si dejábamos que la cuantía de las apuestas fuera totalmente aleatoria, os valores de las ganancias eran desorbitados: por eso hemos decidido que cada apuesta tenga una cuantía menor o igual que 100.

Tuvimos un pequeño problema usando semáforos entre threads y, como andábamos cortos de tiempo y para no tener que gastar mucho tiempo aprendiendo a usar la solución sugerida por google (pthread_mutex), decidimos fabricar nuestros propios semáforos mediante mensajes: protegemos la informacion de las apuestas que cada ventanilla va actualizando con la recepcion bloqueante (down) y el envío (up) de un mensaje.

Con respecto a la simulación de la carrera, el enunciado pide enviar una señal a los caballos para comenzar la simulación (después de haberles enviado la información acerca de la posición del caballo

mediante pipes). Como este mismo pipe sirve para notificar a los caballos del comienzo de la simulación, hemos decidido omitir el envío de la señal.

Una vez iniciada la carrera el programa maneja de forma éxitosa tanto el caso en que se le envía una señal como el caso en el que finaliza de forma normal.

NOTA: Hemos adjuntado el fichero “salidad.txt” con un ejemplo de salida del programa con un menor tiempo de espera al principio y parámetros de entrada bajos para facilitar la observación de la ejecución de la carrera.