Miguel Ángel Sánchez
Juan Velasco

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

Práctica 3: Memoria

Date: 17/04/18

Ejercicio 2

En este programa se crean tantos procesos hijos como se pasen como argumento de entrada y cada hijo es el encargado de dar de alta a un cliente. Para dar de alta a un cliente primero se mira en la memoria compartida cual es el id que le toca, luego se obtiene su nombre por teclado y acto seguido se añade a la memoria compartida.
Finalmente es el padre quien, al recibir la señal que le manda el hijo tras dar de alta a un cliente, lee la información del cliente y la imprime por pantalla.

El problema de este ejercicio está en que no se protege la memoria compartida y se produce lo que se llama condición de carrera. Según cómo de rápidos sean unos procesos u otros podría haber diferentes resultados. A pesar de que a nosotros experimentalmente siempre nos sale lo mismo puesto que no somos lo suficientemente rápidos escribiendo con el teclado, esta no es una buena técnica de programación. Para arreglarlo habría que incluir unos semáforos que protegiesen el acceso a la memoria compartida y esto es precisamente lo que hemos hecho en el ejercicio2_solved.c

Ejercicio 3

De este ejercicio hay poco que comentar puesto que no se usa nada diferente a lo usado en el ejercicio 2. Es un programa que implementa el problema del productor consumidor. Para ello utilizamos memoria compartida y semáforos al igual que en el ejercicio anterior.

Ejercicio 4

En este programa se pide que se invoquen a dos hilos: el primero escribe en un fichero números aleatorios y el segundo lee este fichero y lo imprime por pantalla cambiando las comas por espacios. Para conseguir esto hemos mapeado el contenido del fichero en memoria usando la función *mmap*.

Ejercicio5

En este ejercicio se pide hacer un programa en el que 3 procesos se van pasando información entre sí a través de una cola de mensajes:
El proceso A va leyendo del primer fichero pasado por parámetro y va mandando la información que lee al proceso B. Éste lee esa información de la cola de mensajes, la transforma y se la pasa al proceso C. Por último, el proceso C lee la información obteniendo el mensaje de la cola de mensajes y la imprime en el segundo fichero pasado como argumento de entrada.

Decisiones de diseño y puntos a comentar:
- Primero aclarar que a pesar de que en el enunciado pone que los trozos del fichero que se leen tienen que ser 16KB hemos supuesto que esto es una errata puesto que se necesitaría un fichero inmensamente grande y hemos hecho el ejercicio leyendo 16B.

- Hemos utilizado el parámetro id de la estructura de mensajes para diferenciar entre dos tipos de mensajes: El de id 1 es el que le pasa el proceso A al proceso B y está todavía sin modificar, mientras que el de id 2 es el que le pasa el proceso B al C y está ya modificado.
- Además, hemos añadido un parámetro a la estructura de mensaje. Este parámetro es *ultimo* y es el método que hemos usado nosotros para ver cuál es el último mensaje. Cuando este parámetro está a 1 significa que ya no va a haber más mensajes partiendo de ese proceso.
- A la hora de hacer la traducción del mensaje hemos tenido en cuenta que de la ‘z’ se pasase a la ‘a’ ya fuese minúscula o mayúscula y hemos mantenido los espacios y los saltos de línea sin cambiarlos.
- Por último, decir que incluimos el archivo "entrada.txt" para probar el correcto funcionamiento del ejercicio.