Andrés Mena Godino y Juan Moreno Díez

MEMORIA PRÁCTICA 1

3)Ejercicio

a) No se puede saber en qué orden se imprimirá el texto, ya que cuando se realiza el fork, no podemos saber si va a ser el padre o el hijo el que llegue primero a su respectiva sentencia condicional. Esto ocurre porque no sabemos a cual de los dos procesos va a dar tiempo el procesador ni cuanto tiempo.

b) Para realizar este ejercicio, necesitamos conocer las funciones getpid(), y getppid(), que lo que te devuelven es el id del proceso y el id del proceso del padre respectivamente.

4)Ejercicio

a) Este código puede dejar procesos huérfanos, ya que el padre no espera a que finalicen todos los procesos hijos que ha lanzado (solo hay una sentencia wait, por lo que espera a un hijo, y luego continua con su ejecución). Para solucionar esto, habría que hacer un mejor uso de la función wait, esperando a que finalicen cada uno de sus hijos una vez se han lanzado.

b) Se realiza un bucle para asegurarnos de que esperamos a que se termine cada uno de los hijos, en caso de que no hubiesen terminado antes.

c) Lo que estamos realizando en este programa, es crear una cascada de procesos. Lo realizamos mediante un bucle que sigue el siguiente algoritmo: Un proceso padre, crea un hijo, y este proceso padre lo único que tiene que hacer es esperar a que termine su hijo de ejecutarse. Sin embargo, cuando se vuelve a entrar en el bucle, lo que antes era un hijo, se convierte en padre porque tiene otro hijo, y tiene que esperar hasta que termine su hijo. Esto se hace de forma recursiva hasta que se han hecho todos los procesos.

5)Ejercicio

a) Cuando ejecutamos el programa, primero se reserva memoria para una cadena de caracteres. A continuación, el padre tiene un hijo (se crea una copia del proceso padre, reservando un espacio de memoria para el hijo). El proceso hijo ejecuta una sentencia en la que se copia nuestra frase “hola” en el string para el que habíamos reservado memoria (la cadena que estamos llenando se encuentra en la zona de memoria que ha reservado el proceso hijo para datos). El proceso padre, intenta imprimir lo que hay en la cadena para la cual hemos reservado memoria al principio del programa, sin embargo, esta cadena está vacía, ya que el proceso hijo había llenado la string en la zona de memoria que ha reservado él mismo, diferente a la del proceso padre.

b) la memoria hay que liberarla en ambos, ya que al crear un proceso hijo se ha reservado una zona de memoria para este, ya que el hijo tiene su propia entrada en el BCP, pilas (de s.o y de usuario), y datos. Por ello, el string esta reservado tanto en el proceso padre como en el hijo.

9)Ejercicio

En este programa en c, realizamos la comunicación entre procesos mediante el uso de tuberías. Primero creamos dos procesos, que se comunicaran mediante una tubería (el hijo escribe un número y el padre lo lee). Nos tenemos que asegurar de que el padre espere al hijo que ha creado. Una vez

ya ha ocurrido esto, el padre crea otro hijo, y en esta ocasión es el padre el que escribe y el hijo el que lee. Nos aseguramos de que no queden huerfanos.

12)Ejercicio

En este programa en c, lo que vamos a hacer es un array de identificadores de hilos y un array de estructuras resultados (en la que almacenamos el número del que queremos calcular su exponencial de base 2). Cada hilo rellenará una de las estructuras de resultados, usando la funcion “void *exponencial (void *arg)” y esto se realizara de forma paralela. Antes de finalizar el programa, nos aseguramos de que todos los hilos han finalizado, ya que si el proceso principal finaliza antes, estos hilos podrían no haber realizado las operaciones todavía.