FSO – Práctica 6 - Cuestiones

Cuestión 1: Analice el código y el resultado de la ejecución y responda

1. ¿Qué variables corresponden a los descriptores de archivo en el código propuesto?
   fda 3
   fdb 4
2. Justifique el número asignado por el sistema a la variable fda

   Dado que los dos primeros descriptores de archivos están ocupados por defecto ->
3. Justifique el número asignado por el sistema a la variable fdb

   se empezarán a añadir números a las variables declaradas (3,4,...)

Cuestión 2: Analice el código y el resultado de la ejecución y responda

1. ¿Qué mensajes se imprimen en la pantalla?
   men1: Writing in descriptor 1 (std_output)
   men2: Writing in descriptor 2 (std_error)
   men4: Writing in descriptor 2 (std_error)
2. Justifique por qué no se imprimen cada uno de los mensajes que faltan

   Porque se ejecuta close(1) y close (2) antes de que se impriman
3. Rellene la tabla de descriptores de archivos abiertos correspondiente a dicho proceso antes del return(0)

0	STD_INPUT
1	
2	
3	
4	

Se han cerrado 0 y 1 con sus llamadas close(fd)

Cuestión 3: Analice el código y el resultado de la ejecución y responda

1. ¿Cuál es el contenido del archivo messages.txt?

   cat messages.txt ->

   parent message
   child message
   parent message
2. Tanto el proceso padre como el hijo han escrito su mensaje en el archivo messages.txt. ¿Qué mecanismos/llamadas lo han hecho posible?

Al declarar la variable fd apuntando al archivo messages.txt permite que los hijos del proceso puedan escribir en el archivo, que son creados con fork, y ambos mensajes del padre se van a escribir si o sí, ya que no dependen del fork

3. Rellene la tabla de descriptores de archivos abiertos correspondiente al proceso padre e hijo antes de ejecutar close(fd);

0	STD_INPUT
1	STD_OUTPUT
2	STD_ERROR
3	*cerrado*
4	

0	STD_INPUT
1	STD_OUTPUT
2	STD_ERROR
3	*cerrado*
4	

Cuestión 4: Analice el código y el resultado de la ejecución y responda

1. Justifique utilizando las instrucciones del código el contenido del fichero output.txt

   El código simplemente escribe en el archivo el mensaje declarado en arch y comprueba que no haya errores en la llamada a la función 2

2. Justifique porque la llamada open() se ha invocado con los flags "O_RDWR|O_CREAT"

   Para que se cree el archivo para leer y escribir en el (ReaD, WRite)

3. Rellene la tabla de descriptores de archivos abiertos correspondiente al proceso justo antes del " if(dup…)"

0	STD_INPUT
1	STD_OUTPUT
2	STD_ERROR
3	fd
4	

4. Rellene la tabla de descriptores de archivos abiertos correspondiente al proceso antes de ejecutar return(0)

0	STD_INPUT
1	STD_OUTPUT
2	STD_ERROR
3	
4	

Cuestión 5: Analice el código y el resultado de la ejecución y responda

Tras la ejecución justifique dónde se ha almacenado el contenido del directorio actual

En el archivo salida.txt gracias al redireccionamiento con ">" del comando $ls -la

Cuestión 6: Analice el código y el resultado de la ejecución y responda

¿Qué ha sido necesario modificar en el código del ejercicio5 para llevar a cabo el ejercicio6?

El modo de apertura del fichero a O_RDONLY y STDOUT_FILENO por STDIN_FILENO

Cuestión 7: Analice el código y el resultado de la ejecución y responda

1. ¿Qué muestra el proceso en la salida estándar?

   Muestra 23 líneas, y con el código configurado tanto para salir por consola como redirigido el resultado es el mismo

Cuestión 8: Basándose en el esquema seguido en el ejercicio 7, desarrolle un programa denominado dos_tubos.c que ejecute la siguiente línea de comandos (Dejadlo en Espacio Compartido).