Que

Siguiendo el ejemplo 8, realiza un programa en C que cree un pipe en el que el hijo envía un mensaje al padre, es decir, la información fluya del hijo al padre.

Para que

Para practicar lo aprendido sobre procesos, así como la creación del pipe (la tubería por donde va a pasar el mensaje) y repasar lo aprendido con fork(). También para utilizar read() y write(), que son las funciones que se utilizan para poder escribir o leer en esa tubería.

Como

Lo primero que hemos hecho es ver qué partes del programa son comunes, las cuales realizaremos antes de dividir el proceso en dos, y cuáles son independientes entre sí.

En primer lugar, he definido todas las variables que se van a utilizar. Más tarde, he hecho un `if` para comprobar si la creación del pipe con la función `pipe(fd)` y su modo de lectura/escritura no da error. Si da error, imprime un mensaje por pantalla y asigna el valor -1 a una variable llamada `'retorno'`, para devolver ese valor más tarde. Luego procede a hacer un `fork()` y, con la ayuda de un switch, ejecutamos la función del padre, del hijo o, si hay algún error, se imprime por pantalla y se guarda -1 en la variable `'retorno'`.

Si no ha habido ningún error, el switch salta al caso en el que el `pid` sea 0, es decir, el proceso hijo. Cierra el descriptor de lectura, escribe el mensaje en el pipe con la función `write(modo, mensaje, longitudMensaje)`, cierra el descriptor de escritura e imprime por pantalla que se ha enviado el mensaje.

Y en el último caso del `switch`, en el `'default'`, donde se ejecuta el proceso padre, primero se espera a que el proceso hijo termine, luego cierra el descriptor de escritura (realmente el orden de estas dos líneas no importa, porque puedes cerrar el descriptor primero y luego esperarte o al revés), lee el mensaje del hijo con la función `read(modo, donde se guarda{buffer}`, tamaño del buffer en bytes), cierra el descriptor de lectura e imprime el mensaje. Fuera del `switch`, la función devuelve la variable `'retorno'`.

Pseudocodigo

```
Inicio del Programa
    Declarar variables:
        fd[2] (arreglo de enteros para el pipe)
        retorno (entero)
        pid (entero para el identificador de proceso)
        mensajeHijo (cadena de caracteres)
        buffer (cadena de caracteres)

    Crear una tubería usando pipe(fd)

    Si la creación de la tubería falla
        Mostrar mensaje de error
        Establecer retorno a -1
    Fin Si

    Crear un proceso hijo usando fork()

    Según el valor de pid:
        Caso -1: (ERROR)
            Mostrar mensaje de error
            Establecer retorno a -1
        Fin Caso
        Caso 0: (Hijo)
            Cerrar el descriptor de lectura del pipe (fd[0])
            Escribir mensajeHijo en el pipe (fd[1])
            Cerrar el descriptor de escritura del pipe (fd[1])
            Mostrar "El hijo envía el mensaje al padre..."
        Fin Caso
        Caso por defecto: (Padre)
            Esperar a que el hijo termine (wait)
            Cerrar el descriptor de escritura del pipe (fd[1])
            Leer del pipe (fd[0]) en el buffer
            Cerrar el descriptor de lectura del pipe (fd[0])
            Mostrar "El padre recibe el mensaje del hijo: " + buffer
        Fin Caso
    Fin Según
    Devolver retorno

Fin del Programa
```

Conclusión

Creo que esta práctica me ha servido bastante para reforzar lo enseñado en clase, ya que en clase no lo entendí del todo, pero gracias a la práctica tengo las ideas un poco más claras.