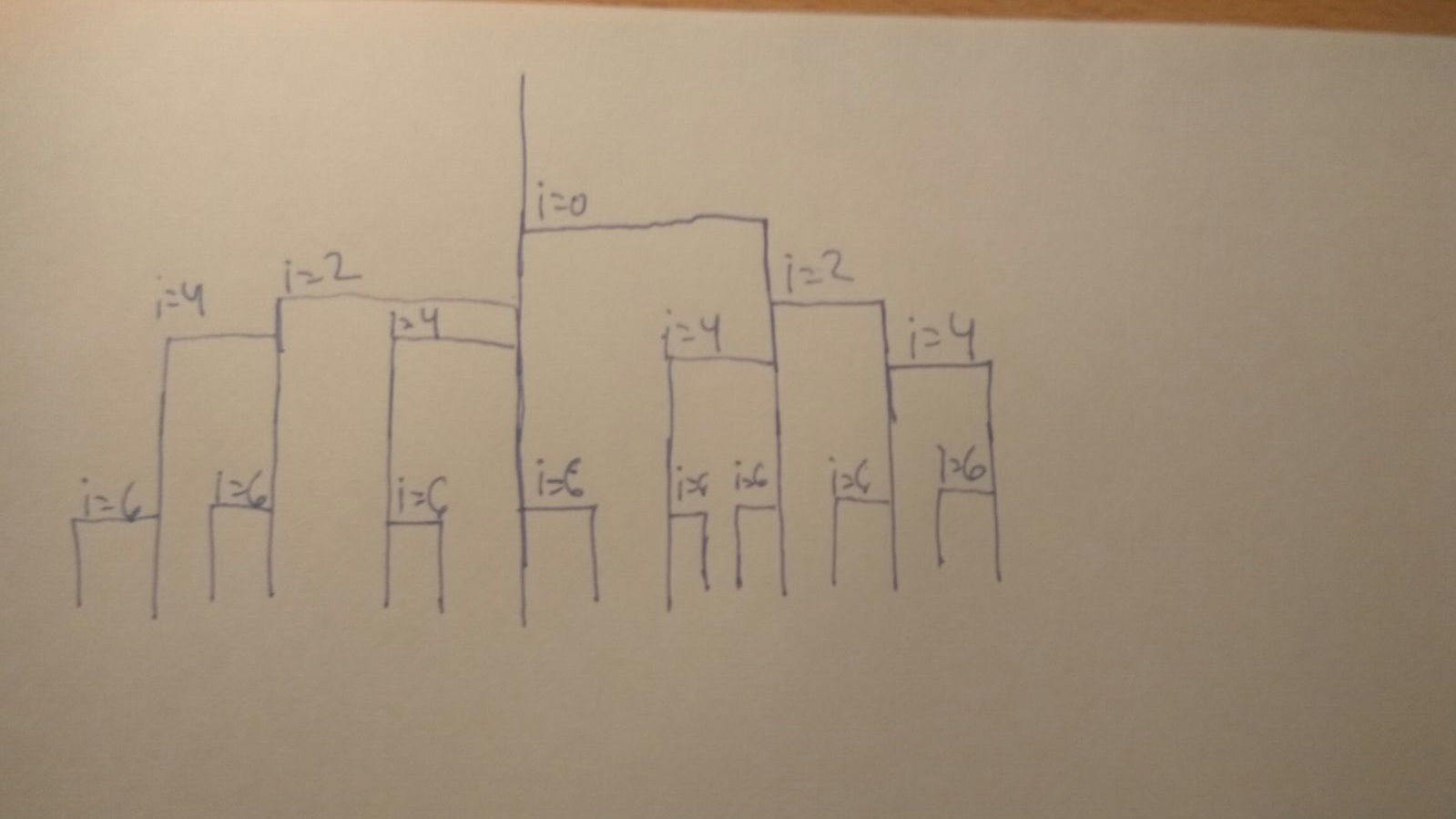
# Práctica 1: Memoria

Date: 08/03/18

**Ejercicio 4**

* **Apartado a**

****

Hemos decidido hacerlo a mano el arbol de procesos. Estos son todos los procesos que se crean. Decir que existe la probabilidad (dado que no hay ningún wait) de que todos los procesos que no sean el proceso padre se queden huerfanos. Si un hijo termina antes que su padre se quedará zombie.

* **Apartado b**

Dado que no sabemos el tiempo de ejecución, un hijo puede acabar antes que su padre (quedando zombie) o un padre puede acabar antes que el hijo (quedando el hijo huérfano). En este segundo apartado todos los procesos realizan un wait. Por eso, los procesos generados en i = 6 no esperaran a nadie y acabaran. Los generados en i = 4 tendran que esperar a su hijo correspondiente. El problema viene con los procesos padre y los generados en i = 0 e i = 2. Estos tienen más de un hijo (4 , 3 y 2 respectivamente) y solo esperaran a uno. Por tanto existe la posibilidad de que el resto acaben huérfanos como hemos explicado antes.

En la salida de ambos apartados modificados es donde podemos ver esto. Si su padre es el proceso 1 serán huérfanos y si no serán recogidos por el padre.

**Ejercicio 5**

* **Apartado a**

Los dos únicos cambios introducidos han sido la comprobación (Ahora se generarán procesos cuando i%2!=0 y no cuando i%2==0), y un break. Este break ha sido añadido debido a que cada proceso solo puede tener un hijo, luego en cada fork que se haga, el padre tiene que salir del bucle, para no hacer más hijos.

* **Apartado b**

Si de nuevo partimos del ejercicio 4modb, ahora hemos realizado tres cambios: la comprobación (i%!2!=0 en lugar de 1%2==0), un break y un waitpid.

Este break ha sido puesto ahora donde el hijo. Esto es debido a que en este ejercicio el padre puede tener varios hijos, pero estos hijos no pueden tener hijos, luego tienen que salir del bucle.

Y el waitpid es para que el padre vaya recogiendo a sus hijos, y no termine sin hacerlo.

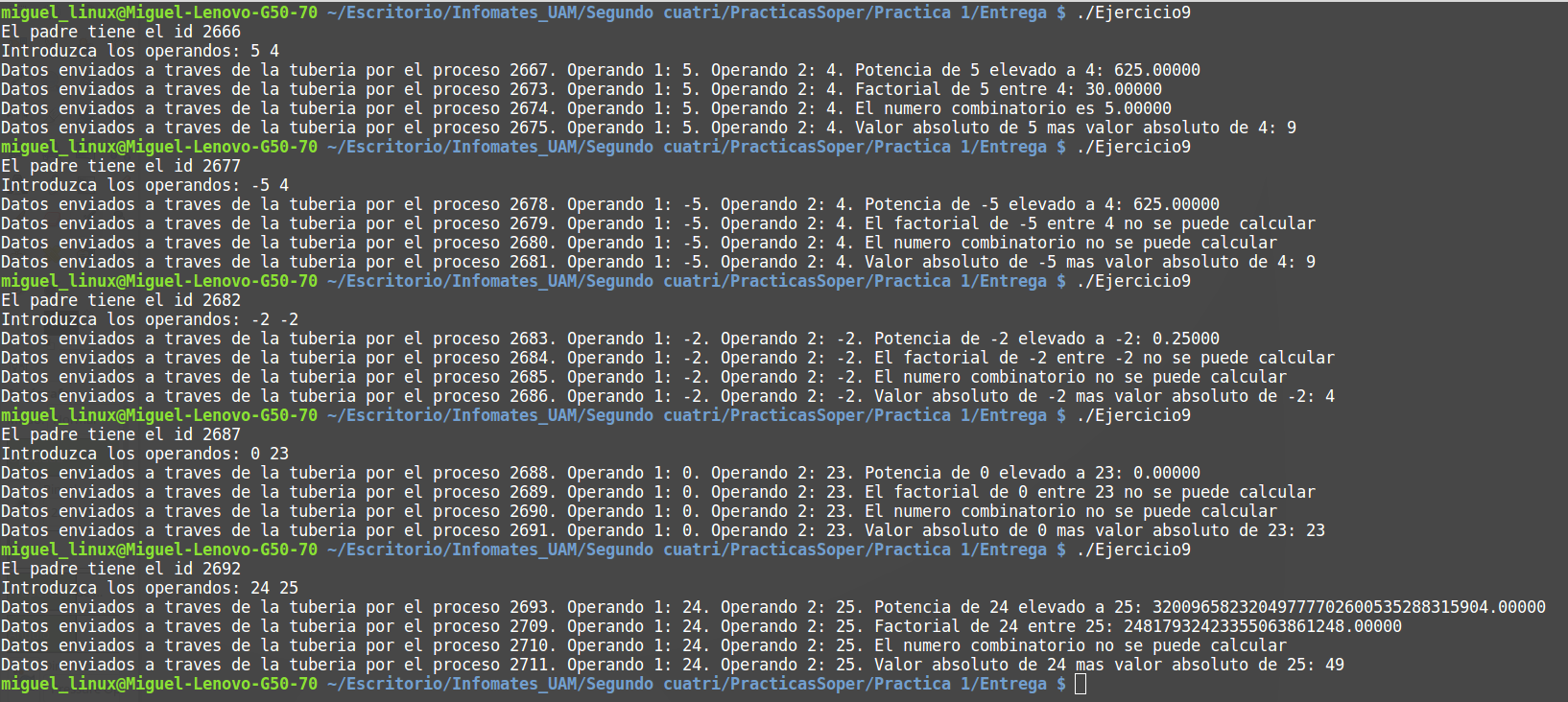
**Ejercicio 6**

El proceso padre no tiene acceso a ese valor.

Al hacer un fork, el proceso hijo copia la información y variables del proceso padre, pero una vez separados todo lo que hagan lo harán independientemente (diferentes regiones de memoria). Esto implica que, si el proceso hijo guarda información en una variable local suya, el proceso padre no tendrá acceso a esta variable.

Ambos procesos tendrán que liberar la memoria, pues a pesar de que la reservó únicamente el padre, al hacer el fork el hijo también reserva esa memoria. Por eso hemos hecho el free para los dos procesos al final del código.

**Ejercicio 9**

****

A continuación comentaremos los distintos casos en los que devolveremos error en cada una de nuestras operaciones (asumiendo que se escriben dos enteros por teclado):

En el caso de la potencia no hay ningún problema. Se pasan dos enteros y se devuelve el primero elevado al segundo.

En la segunda operación nos aseguramos de que el primer entero sea positivo para poder calcular correctamente el factorial, puesto que no existe el factorial de un número negativo; y de que el segundo entero no sea 0 para no dividir entre 0.

En la tercera operación, la del número combinatorio, nos aseguramos de que ninguno de los dos parámetros sea menor o igual que cero, y también de que el segundo no sea mayor que el primero.

Y por último, para el de los valores absolutos se comprueba si alguno de los enteros es negativo para cambiarlo a positivo.

En caso de no ser un parámetro correcto la función devuelve “No se ha podido calcular”.

**Ejercicio 12**

Es más rápido el ejercicio12b, la opción en la que se usan hilos. Esto es claro debido a que el tiempo de creación y terminación de los hilos es más corto que el de los procesos.

12 a → 102738 milisegundos

12 b → 102054 milisegundos

**Ejercicio 13**

Compartir información entre hilos es mucho más sencillo que entre procesos. Esto es debido a que los hilos comparten la memoria y los recursos del proceso al que pertenecen, luego no es necesario el uso de pipes. Simplemente modificando el valor de una variable global del proceso pueden pasarse información.

**Nota:** todos los comentarios respectivos al código y a la implementación concreta de cada código se encuentra en el propio código y en doxygen.