DOCUMENTACIÓN PRÁCTICA 1
SISTEMAS OPERATIVOS

Ejercicio 2:

En este ejercicio creamos un programa que crea 4 procesos hijos. Estos a su vez imprimirán un mensaje, dormirán un tiempo y volverán a imprimir otro mensaje. Mientras tanto, el padre duerme un tiempo mucho menor y les manda una señal de terminación. Por lo tanto, los hijos son matados mientras esperan, por lo que no llegan a imprimir el segundo mensaje

Salida de ejecución:

```
$ ./ejercicio2
Soy el proceso hijo <27402>
Soy el proceso hijo <27403>
Soy el proceso hijo <27406>
Soy el proceso hijo <27409>
```

Ejercicio4:

En este ejercicio creamos un programa que crea un proceso hijo, que empieza a realizar un trabajo (imprimir un mensaje y esperar). Tras realizarlo 10 veces, le pide el relevo al padre, mediante señales. Una vez el padre lo recibe, mata el proceso y crea otro nuevo, un máximo número de veces, tras las que termina su ejecución.

Salida de ejecución:

```
$ ./ejercicio4 3
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
Soy <27497> y estoy trabajando
```

```
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27499> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando
Soy <27500> y estoy trabajando

Terminado
```

Se puede ver que los hijos imprimen su mensaje 11 veces (una vez más de las mínimas), puesto que al imprimirlo 10 veces mandan una señal al padre, pero su terminación no es inmediata. Tras tres procesos diferentes (lo indicado en los argumentos de llamada) la ejecución termina

Ejercicio6:

En este ejercicio creamos dos programas (ejercicio6a y ejercicio6b), con una funcionalidad muy similar: hacer un bucle imprimiendo mensajes, pero con la diferencia de que en el ejercicio6a se bloquean las señales antes de cada bucle, y se desbloquean después. Esto

hace que la señal que finaliza la ejecución del programa no sea procesada hasta después de terminar el bucle. Esto se ve muy claramente en las salidas de los programas:

```
$ ./ejercicio6a
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
```

```
$ ./ejercicio6a
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
3
Soy <27631> y he recibido la
señal SIGTERM
```

(Para esta salida pusimos 39 segundos, para visualizarlo mejor, puesto que, si usamos 40, es múltiplo de 5 y coincide con el final de un bucle. Si ponemos 39 en el ejercicio6a, se imprime exactamente lo mismo)

Se puede observar que el ejercicio6b termina sin llegar a imprimir un bucle entero

Ejercicio8:

En este ejercicio se nos pide crear el TAD semáforo, que implementa las funciones típicas de los semáforos vistas en la teoría, con las funciones que C tiene primitivas de semáforos.

El fichero semaforos.h nos lo daban en la documentación de la práctica y nosotros hemos tenido que, a raíz de un ejemplo de manejo de las funciones primitivas de C de semáforos, implementar las funciones pedidas.

No tuvimos especial dificultad con ello.

Ejercicio9:

Este ejercicio ha sido el más largo de toda la práctica y ha supuesto un gran reto, ya que se nos pide implementar el funcionamiento de un supermercado usando procesos hijos para cada una de las cajas y el proceso padre del programa es el "encargado" del supermercado.

La comunicación entre procesos ha sido especialmente difícil en este ejercicio, ya que cada hijo tenía que llamar a su padre para que le retirase en dinero cuando llegase a 1000 euros, y también cuando acababa el turno para que le retirase todo el sueldo restante.

Esto lo hemos hecho usando señales. La primera dificultad que tuvimos fue que usando las señales SIGUSR1 y SIGUSR2 no funcionaba, ya que estas señales no eran apilables y el padre no recibía bien las señales mandadas por el hijo, puesto que descartaba algunas de ellas.

Esto lo solucionamos usando las señales SIGRTMIN y SIGRTMIN+1, las cuales encontramos tras cierta investigación por diversas webs.

Tuvimos también cierta complejidad al leer y escribir en los ficheros, ya que tuvimos que coordinarlo todo correctamente con los semáforos y hacer las escrituras y lecturas correctamente.

A causa de la complejidad del ejercicio, tuvimos bastantes dificultades debuggeando y tras bastantes horas conseguimos que funcionase.

Nos ha resultado un ejercicio de una complejidad superlativa en comparación con el resto, pero sin embargo nos ha servido para entender bien cómo es un programa realmente complejo en el que se usa comunicación entre procesos.