MEMORIA PRACTICA 2 SISTEMAS OPERATIVOS 2016-2017

Ejercicio 3:

En este ejercicio realizamos las mismas operaciones con un proceso hijo y con un hilo para comprobar si hay una diferencia de tiempos.
Para medir la diferencia de tiempos utilizaremos la estructura timeval y la función gettimeofday(). Tras ejecutar los apartados a y b comprobamos que los hilos son más rapidos (en torno a 10 segundos para una muestra de 10000 primos):

```
nacho@Superordenador:~/Escritorio/Uni/SEGUNDO/Soper/Practica2$ ./ejercicio3a 100
00
El programa ha tardado 55.632541 segundos en calcular 10000 primos.
nacho@Superordenador:~/Escritorio/Uni/SEGUNDO/Soper/Practica2$ ./ejercicio3b 100
00
El programa ha tardado 45.523893 segundos en calcular 10000 primos.
```

Ejercicio 4:

En este ejercicio realizaremos un programa con dos hilos que comparten datos entre ellos. Para ello utilizaremos la siguiente estructura:

```
struct _Hilo
{
	Hilo *otro; /*!<Estructura del otro hilo*/
	int progreso; /*!<Progreso del cálculo*/
	int numero; /*!<Número del hilo*/
	int mul; /*!<Multiplicador aplicado a el hilo*/
	pthread_t p; /*!<El hilo*/
	int **matriz; /*!<matriz a calcular*/
	int *dim; /*!<Dimensión de la matriz*/
};
```

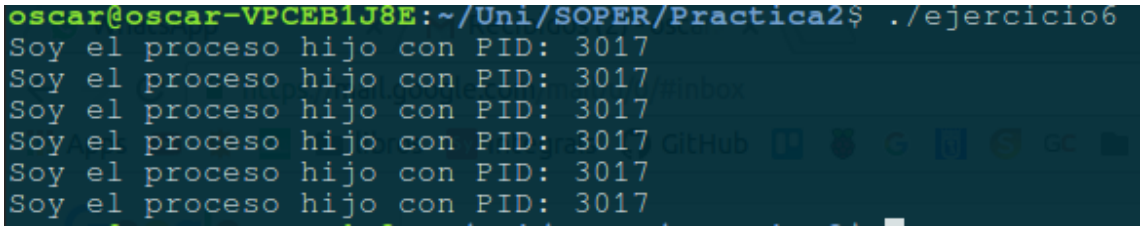
En ella almacenamos la estructura del otro hilo y los datos necesarios del hilo, como el progreso de la multiplicación de matrices, la matriz original y la dimensión de la matriz.

Gracias a esta estructura podemos realizar el cálculo:

```
oscar@oscar-VPCEB1J8E:~/Uni/SOPER/Practica2$ ./ejercicio4
Introduzca dimension de la matriz cuadrada:
3
Introduzca matriz 1:
1 2 3
4 5 6
7 8 9
Introduzca matriz 2:
12 13 14
15 16 17
18 19 20
Introduzca multiplicador 1:
10
Introduzca multiplicador 2:
100
Hilo 1 multiplicando fila 0 resultado: 10 20 30 - el Hilo 2 va por la fila 0
Hilo 2 multiplicando fila 0 resultado: 1200 1300 1400 - el Hilo 1 va por la fila 1
Hilo 1 multiplicando fila 1 resultado: 40 50 60 - el Hilo 2 va por la fila 1
Hilo 2 multiplicando fila 1 resultado: 1500 1600 1700 - el Hilo 1 va por la fila 2
Hilo 1 multiplicando fila 2 resultado: 70 80 90 - el Hilo 2 va por la fila 2
Hilo 2 multiplicando fila 2 resultado: 1800 1900 2000 - el Hilo 1 va por la fila 3
```

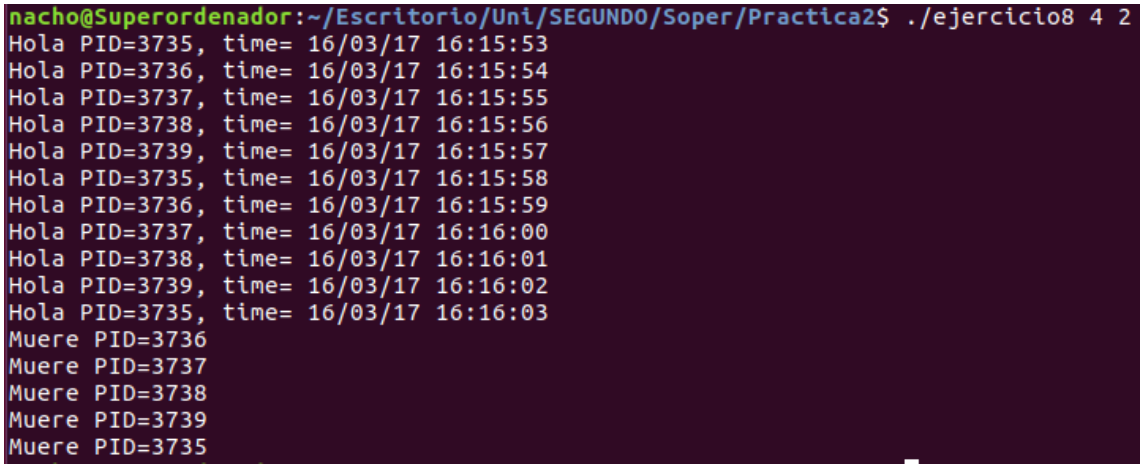
Ejercicio 6:

En este ejercicio utilizaremos una señal para detener un hijo. Para ello guardaremos el pid del hijo en una variable en el padre y dividiremos el código en un bucle del hijo que cada 5 segundos imprima por consola. Cuando pasan 30 segundos, el padre “matará” al proceso hijo con la señal SIGKILL y terminará.



Ejercicio 8:

En este ejercicio creamos un bucle que crea los **n** hijos, posteriormente realizamos el paso de señales SIGUSR1 entre ellos durante **v** vueltas, una vez terminan, los procesos se irán pasando una señal SIGTERM, muriendo después de pasarla. Por ello obtenemos la siguiente impresión por pantalla, donde el último que termina es el padre:



Ejercicio 10:

En este ejercicio decidimos implementar una máscara para capturar señales. El proceso A tiene una sección crítica (Escritura de fichero), por lo que, como el proceso B tiene que leerlo, paramos el proceso B hasta que A termine su sección crítica para darle paso a B. Esto lo realizamos por medio de las señales USR1 y USR2.

Para comprobar si A está activo utilizamos waitpid con la opción WNOHANG, para que el padre no se bloquee y continúe con la ejecución.