SISTEMAS OPERATIVOS: Informe práctica 2

Ejercicio 3:

En este ejercicio se pedía analizar el comportamiento ante una misma situación de hilos y de procesos. Esta situación es el cálculo de los n primeros primos siendo n el valor pasado por pantalla.

Como se puede observar en la captura de la ejecución, cuando se calculan un número de primos muy pequeño los tiempos que salen son esperados, tarda más en crear procesos que en crear hilos (tal y como se explicó en las clases de teoría) ya que los hilos son menos costosos de crear y de eliminar. Sin embargo, con la entrada de 10.000 primos observamos que los resultados son totalmente contrarios (tarda más con hilos que con procesos). Esta situación es debida a que el planificador reparte por grupos y en los procesos todos tendrán el mismo tiempo mientras que los hilos se reparten los tiempos del mismo proceso teniendo por tanto menos tiempo para la realización de la operación pedida. Por tanto, al ejecutar una función que requiere más tiempo del que le damos a cada hilo,el planificador cambia de proceso, haciendo que el proceso de los hilos tarde más en realizar sus operaciones.

Salida ejercicio3a.c y ejercicio3b.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3a 100
Tiempo de ejecucion: 0.030968s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3b 100
Tiempo de ejecucion: 0.008756s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3a 1000
Tiempo de ejecucion: 0.085355s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3b 1000
Tiempo de ejecucion: 0.05795s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3a 10000
Tiempo de ejecucion: 0.637663s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3b 10000
Tiempo de ejecucion: 0.650863s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3a 100000
Tiempo de ejecucion: 16.7396s
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio3$ ./ejercicio3b 100000
Tiempo de ejecucion: 17.2703s
```

Los ficheros ejercicio3a.c y ejercicio3b.c contienen los códigos de procesos y de hilos respectivamente. En ambos ejercicios, realizamos comprobaciones del paso de parámetros como que lo que se pase sea un número y que sea solo un único parámetro.

Ejercicio 4:

En este ejercicio se pedía la implementación de dos hilos para que cada uno de ellos calculase un producto escalar pedido al usuario de una matriz cuadrada también pedida al usuario distintos entre sí (es decir, se piden dos matrices y dos escalares). El objetivo del ejercicio es ver el paralelismo de la ejecución de ambos hilos introduciendo si hiciera falta para ello tiempos de espera.

Para ello, la implementación que hemos realizado es la creación de una estructura que nos facilitará después la codificación para el apartado b en la que guardaremos toda la información

que necesitamos para que el hilo realice la operación. En la implementación tanto del apartado b como del a usamos a parte de las estructuras, una función main en la que se le pide al usuario los parámetros necesarios y se comprueban todos los errores pedidos así como aquellos que pueden ocurrir en el manejo de hilos; y una segunda función, la que realizarán los hilos para el cálculo pedido, en la que incluimos una llamada a la función usleep para observar mejor el paralelismo de la ejecución. Además hemos decidido incluir una tercera función que permitirá comprobar con exactitud que los parámetros pasados son los adecuados.

En la salida del apartado a, se produce la salida esperada y se pasan parámetros por pantalla no correctos para permitir observar la robustez del programa ante errores del usuario.

Salida de ejercicio4a.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio4$ ./ejercicio4a
Introduzca dimension de la matriz cuadrada:
5
La dimensión ha de ser un entero positivo menor que 4:
3
Introduzca multiplicador 1:
6
Introduzca multiplicador 2:
3
Introduzca matriz 1:
1 2 3 4 5 6 7 8
No se han introducido suficientes números para la primera matriz. Vuelva a intentarlo:
1 2 3 4 5 6 7 8 9
Introduzca matriz 2:
9 8 7 6 5 4 3 2 1
Hilo 2 multiplicando fila 0 resultado 27 24 21
Hilo 1 multiplicando fila 0 resultado 6 12 18
Hilo 2 multiplicando fila 1 resultadoHilo 1 multiplicando fila 1 resultado 24 30 36
 18 15 12
Hilo 1 multiplicando fila 2 resultado 42 48 54
Hilo 2 multiplicando fila 2 resultado 9 6 3
```

En el apartado b, se pide pensar cómo se podría intercambiar información entre dos hilos y para ello lo único que hemos tenido que modificar con respecto al apartado anterior es la inclusión en la estructura de cada hilo, la estructura del otro hilo para poder así tener acceso desde uno al otro y un parámetro fila para poder llevar la cuenta de las filas del otro hilo.

En la salida de este apartado, se produce de nuevo la salida esperada y esta vez al ser el mismo código no probamos la robustez del programa que sigue siendo la misma.

Salida de ejercicio4b.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio4$ ./ejercicio4b
Introduzca dimension de la matriz cuadrada:
3
Introduzca multiplicador 1:
3
Introduzca multiplicador 2:
6
Introduzca matriz 1:
1 2 3 4 5 6 7 8 9
Introduzca matriz 2:
9 8 7 6 5 4 3 2 1
Hilo 1 multiplicando fila 0 resultado 3  6  9 - el Hilo 2 va por la fila 0
Hilo 2 multiplicando fila 0 resultado 54  48  42 - el Hilo 1 va por la fila 1
Hilo 1 multiplicando fila 1 resultado 12  15  18 - el Hilo 2 va por la fila 1
Hilo 2 multiplicando fila 1 resultado 36  30  24 - el Hilo 1 va por la fila 2
Hilo 1 multiplicando fila 2 resultado 21  24  27 - el Hilo 2 va por la fila 2
Hilo 2 multiplicando fila 2 resultado 18  12  6 - el Hilo 1 ha acabado
```

Ejercicio 6:

En este ejercicio se pide que el hijo mande información sobre su PID por pantalla cada cierto tiempo mientras que el padre no lo mate. El objetivo fundamental del ejercicio es aprender el manejo de señales.

En nuestra implementación, decidimos pasar al hijo la primera señal de la lista que hacía que el proceso finalizase (SIGHUP). Su salida es la esperada.

Salida de ejercicio6.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/Ejercicio6$ ./ejercicio6
Soy el proceso hijo con PID: 7699
Soy el proceso hijo con PID: 7699
Soy el proceso hijo con PID: 7699
Soy el proceso hijo con PID: 7699
Soy el proceso hijo con PID: 7699
Soy el proceso hijo con PID: 7699
```

Ejercicio 8:

En este ejercicio se pedía realizar un programa muy extenso que pide al usuario dos parámetros: uno será el número de procesos que creará el programa en serie y otro , el número de veces que se pase el testigo. Tras ello, el programa ejecuta una rutina especificada en la práctica con detenimiento y que consiste en que desde el último hijo hasta el proceso raíz se vayan pasando señales de padres a hijos. Este protocolo comenzará con el envío de la señal al proceso padre raíz por parte del último de los hijos. En segundo lugar, el proceso ejecutará esta rutina el número de veces indicado por el parámetro pasado V. Y por último, cuando la raíz detecte que ya se han producido las V vueltas enviará al proceso hijo una señal de terminación y así hasta llegar al último hijo, que matará al proceso raíz terminando así el programa.

Por otro lado todo esto se realizará con la ayuda del código facilitado para imprimir la hora.

A la hora de implementar este ejercicio, decidimos no hacer uso de variables globales y por tanto tener un manejador sencillo, el problema que surgió entonces era como pasar información entre padres e hijos y decidimos hacer un array para ello.

Salida de ejercicio8.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/A Entregar$ ./ejercicio8 5 2
Hola PID=5761,time=16/03/17 01:14:10
Hola PID=5762,time=16/03/17 01:14:12
Hola PID=5763,time=16/03/17 01:14:14
Hola PID=5764,time=16/03/17 01:14:16
Hola PID=5765,time=16/03/17 01:14:18
Hola PID=5766,time=16/03/17 01:14:20
Hola PID=5761,time=16/03/17 01:14:25
Hola PID=5762,time=16/03/17 01:14:27
Hola PID=5763,time=16/03/17 01:14:29
Hola PID=5764,time=16/03/17 01:14:31
Hola PID=5765,time=16/03/17 01:14:33
Hola PID=5766,time=16/03/17 01:14:35
Muere 5762
Muere 5763
Muere 5764
Muere 5765
Muere 5766
Muere 5761
```

En la realización de este informe y tras comentarlo con el resto de compañeros, observamos que había una segunda opción, con las variables globales. Tras preguntarle al profesor de teoría y a nuestro correspondiente profesor de prácticas y darnos ambos su aceptación, decidimos hacer una segunda versión con variables globales y un par de manejadores para las señales usadas. Su resultado es exactamente el mismo y es el que se muestra a continuación.

Salida de ejercicio8b.c:

```
andressp05@andressp05-X556UJ:~/Escritorio/Soper/Practica 2/A Entregar$ ./ejercicio8b 5 2
Hola PID=5799,time=16/03/17 01:15:12
Hola PID=5800,time=16/03/17 01:15:14
Hola PID=5801,time=16/03/17 01:15:16
Hola PID=5802,time=16/03/17 01:15:18
Hola PID=5803,time=16/03/17 01:15:20
Hola PID=5804,time=16/03/17 01:15:22
Hola PID=5799,time=16/03/17 01:15:24
Hola PID=5800,time=16/03/17 01:15:26
Hola PID=5801,time=16/03/17 01:15:28
Hola PID=5802,time=16/03/17 01:15:30
Hola PID=5803,time=16/03/17 01:15:32
Hola PID=5804,time=16/03/17 01:15:34
Muere 5800
Muere 5801
Muere 5802
Muere 5803
Muere 5804
Muere 5799
```

Ejercicio 10:

En este ejercicio se pedía implementar un programa con un proceso padre B y un proceso hijo A. Lo que realiza el proceso hijo A es seleccionar aleatoriamente palabras de la cadena: “EL PROCESO A ESCRIBE EN UN FICHERO HASTA QUE LEE LA CADENA FIN”. Una vez leída la palabra la escribirá en un fichero y espera al padre, en caso de que la palabra que lea sea “FIN” el proceso A la escribe en el fichero y finaliza. Por otro lado, el proceso A espera a que el hijo escriba y una vez escrito, lee esa palabra del fichero y, en caso de que la palabra leída sea “FIN” interpreta que el proceso hijo B ha terminado y lo vuelve a crear. La condición de salida del bucle es la lectura de 50 cadenas.

Por otro lado, y aunque no se pedía, como el ejercicio se encontraba tras la explicación de máscaras hemos decidido incluir diversas funciones sobre el manejo de máscaras para comprobar su correcto funcionamiento. Hemos decidido enmascarar todas las señales excepto SIGUSR1 SIGUSR2.

NOTA: esta es la interpetración que hemos realizado ya que el enunciado dejaba muchas dudas.

Justo abajo se encuentran las dos salidas que se producen, por un lado la cadena.txt generada durante la ejecución y por otro lado la salida de la ejecución del ejercicio en sí. En esta última es donde se aprecia que cuando el hijo lee fin este termina y el padre creará un nuevo hijo manifestándose esto en el diferente PID del hijo siguiente.

Fichero cadena.txt generado:

EL LEE CADENA EN PROCESO PROCESO FIN LEE PROCESO EN UN PROCESO
CADENA ESCRIBE HASTA HASTA UN LA EN LEE EN EL PROCESO PROCESO
FICHERO CADENA QUE EN A UN CADENA A LEE LEE ESCRIBE LEE FIN HASTA
CADENA QUE LA PROCESO ESCRIBE A FIN FICHERO FIN HASTA LEE A

Salida de ejercicio10.c:

```
> ./ejercicio10

HIJO 7419 escribe EL:
Padre lee cadena 1: EL

HIJO 7419 escribe LEE:
Padre lee cadena 2: LEE

HIJO 7419 escribe CADENA:
Padre lee cadena 3: CADENA

HIJO 7419 escribe EN:
Padre lee cadena 4: EN

HIJO 7419 escribe PROCESO:
Padre lee cadena 5: PROCESO

HIJO 7419 escribe PROCESO:
Padre lee cadena 6: PROCESO

HIJO 7419 escribe FIN:
Padre lee cadena 7: FIN

HIJO 7420 escribe LEE:
Padre lee cadena 8: LEE

HIJO 7420 escribe PROCESO:
Padre lee cadena 9: PROCESO
```