USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

SISTEMAS OPERATIVOS I

Práctica 4: Utilización de hilos

Objetivo

Introducción al manejo de hilos y a su funcionamiento.

Información útil para la práctica

Los hilos de un proceso comparten el espacio de memoria: las variables **globales** se comparten entre los hilos, mientras que las **locales** al hilo principal o a los otros hilos que se crean no se deben compartir, son locales a cada hilo. De todas formas, cualquier hilo puede acceder a cualquier variable del proceso a través de su dirección (puntero).

- Lee el manual de uso de *pthread_ create* para crear hilos y de *pthread_ exit* para finalizarlos.
- Lee el manual de uso de la llamada al sistema *pthread_ join* para esperar a la finalización de un hilo en el hilo creador.
- Al compilar acuérdate de incluir la opción de compilación: **-lpthread**.

Enunciado

1. Haz un programa para comprobar que tanto las variables globales como las locales a los hilos son accesibles por todos ellos. Comprueba que si un hilo cambia una de esas variables, los demás "ven" dicho cambio.
2. Haz un programa que cree tres hilos y que el último de ellos cree un proceso hijo con *fork*. Comprueba con el comando *ps* cuantos hilos tiene ese proceso hijo, es decir, si se crea también con tres hilos. Comprueba que ocurre con este proceso hijo cuando termina el hilo desde el que se ha creado (sin cambiar la imagen del proceso). En el código debes garantizar que cuando el tercer hilo ejecuta el *fork*, los demás no han acabado.
3. Haz un programa con tres hilos (además del hilo principal) de modo que uno de ellos ejecute *exit* en lugar de *pthread_ exit*, antes de que los otros hilos terminen y comprueba que efecto tiene este *exit* sobre los demás hilos. Debes garantizar que los hilos no finalizan previamente a la ejecución de dicho *exit*.
4. Haz un programa para comprobar el efecto de la función *sched_ yield* (*pthread_ yield* puede usarse en versiones antiguas de la biblioteca de C de GNU). Una forma sencilla de hacer esto es crear un programa con varios hilos, de modo que algunos sean mucho más generosos que los demás liberando la CPU muy frecuentemente. Se trataría de comprobar que cuando se hace la llamada a esa función, los hilos generosos acaban más tarde que los que no lo son.

5. **ENTREGABLE** Haz un programa que aproxime numéricamente el valor del problema de Basilea para n desde 1 hasta mil millones ($1e9$) usando varios hilos.

   El problema de Basilea consiste en encontrar la suma exacta de los inversos de los cuadrados de los enteros positivos, esto es, la suma exacta de la serie infinita:

$$\sum_{n=1}^{\infty} \frac{1}{(n^2)} = \frac{\pi^2}{6} \tag{1}$$

   Para obtener esta aproximación, escribe un programa que cumpla con las siguientes especificaciones:

   (a) Todos los cálculos se realizan en doble precisión.

   (b) El número de hilos H debe estar definido como una constante en el código. Busca información sobre el número de cores de tu equipo[1] para conocer las limitaciones hardware de tu sistema.

   (c) El j-ésimo hilo H_j realiza el cálculo de 10 iteraciones consecutivas, haciendo un reparto cíclico de las iteraciones del sumatorio entre todos los hilos. Por ejemplo, si $H = 4$, el hilo H_2 hace el cálculo de las iteraciones $[11..20, 51..60, 91..100, ...]$.

   - Empieza contando las iteraciones desde 1 para evitar división por cero.
   - Comprueba que el número total de iteraciones es correcto, por ejemplo, cambiando el sumatorio de la serie por el sumatorio del número de iteraciones. Después de comprobarlo, comenta el código para dejarlo en el entregable sin que afecte al resto de los apartados.

   (d) Cada uno de los hilos hace su cálculo sobre una variable sumatorio diferente para evitar carreras críticas. Todos los hilos trabajan de forma concurrente.

   (e) El hilo principal debe esperar a que acaben todos los hilos para calcular el sumatorio total sumando las contribuciones parciales de cada hilo para obtener el valor final de la serie.

   (f) Además, el hilo principal hace el cálculo completo de forma secuencial una vez haya calculado el sumatorio total de los hilos, y muestra los resultados de la versión paralela y secuencial, así como la diferencia entre ambas versiones y la diferencia con respecto al valor exacto ($\frac{\pi^2}{6}$).

   (g) Consulta el monitor del sistema para comprobar que todos los cores necesarios están trabajando concurrentemenete al ejecutar el código y comprueba cuanto tiempo se ahorra frente a la versión secuencial en función del número de hilos empleado.

   (h) Comprueba qué sucede si el número de hilos empleado es superior al de cores de tu equipo. Incluye tus observaciones como un comentario al principio del propio código entregado.

[1] Puedes buscar información de la CPU por ejemplo en https://en.wikichip.org/wiki/WikiChip.