Lab01 - OS - 2020-II

{TA:martin.carrasco, Prof: jgonzalez} @utec.edu.pe

October 2, 2020

1 Indicaciones

1. Escribir un informe explicando de forma breve y directa la solución de cada ejercicio. Para justificar la respuesta, según sea necesario mostrar: demostraciones, pseudocódigo (o snippets de código), diagramas, tablas, etc..
2. Separar en carpetas para cada ejercicio **con los nombres:** Ejercicio 1, Ejercicio 2, Ejercicio 3, Ejercicio 4. Subir en un archivo comprimido (.zip) en Gradescope **solamente**: 1) carpetas con archivos en C y el Makefile (el comando para compilar); e 2) informe en .pdf (fuera de las carpetas).
3. Fecha de **entrega**: `Verificar Sección de Tareas Lab01 de Canvas`.
4. Las entregas son realizadas **solamente por Gradescope.**
5. Cualquier intento de plagio parcial o total es sancionado por UTEC en todas sus atribuciones.

2 Ejercicios

1. Para calcular el valor de π se puede usar la técnica Monte Carlo la cual consiste en:
   (a) Imaginar que se tiene un círculo dentro de un cuadrado (asumir el $radio = 1$).
   (b) Generar una serie de puntos random como coordenadas simples `(x,y)`. Estos puntos deben estar en los límites del cuadrado y algunos estarán dentro del círculo.
   (c) Estimar:

$$\pi = 4 * (numero\_puntos\_en\_circulo)/(numero\_total\_de\_puntos)$$

.

Escribir una versión multithreaded del algoritmo en C (usar POSIX), que cree una thread separada para generar un número de puntos random. **Nota:** generar números random tipo double entre -1 y +1. La thread contará el número de puntos que caen dentro del círculo y guardará el resultado en una variable global. Solamente cuando la thread child termine, la thread parent calculará e imprimirá el valor estimado de π. Se recomienda experimentar con el número puntos random generados, mientras mayor sea el número la aproximación a π será mayor.

2. Extender el programa anterior para n threads, cada una de las cuales genera puntos random y determina si estos están dentro del círculo. Cada thread debe actualizar la cuenta global de todos los puntos que caen en el círculo. Usar mecanismos de exclusión mutua POSIX mutex_lock para proteger la variable global compartida de race conditions en la actualización de valores.

3. La secuencia Fibonacci es la series de números 0, 1, 1, 2, 3, 5, 8, ... Formalmente se expresa como:

$$fib_0 = 0$$

$$fib_1 = 1$$

$$fib_n = fib_{n-1} + fib_{n-2}$$

Escribir un programa multithreaded en C (usar POSIX) que genere la secuencia Fibonacci con las siguientes características:

- El programa recibe como argumento la cantidad de números Fibonacci a generar.
- Luego el programa crea una thread separada que genera los números Fibonacci, colocándolos como datos que pueden ser compartidos por threads (ejemplo: un array es la forma más simple y conveniente para la estructura de datos).
- Cuando la thread child finaliza ejecución, la thread parent imprime la secuencia generada por la thread child. La thread parent debe esperar que la thread child finalice para poder imprimir.

4. Modificar el programa anterior para que la salida de números Fibonacci pueda ser accesada por la thread parent una vez que los valores son generados en la thread child, en lugar de esperar a que la thread child termine. Realizar dos implementaciones POSIX: 1) con semáforos, y b) con futex.

3 Referencias

1. **Tutorial Reference:** Pthread tutorial.

2. Libros de referencia y material de clase (ver Sílabo y Canvas).