

**DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN**

# Sistemas Operativos

Laboratorio 4

Threads y Concurrencia

# INTRODUCCIÓN

Estamos en una era de programación multiprocesador o multinúcleo, donde cada computadora está equipada con múltiples procesadores, cada uno de los cuales a su vez puede tener varios núcleos. Para explotar completamente la capacidad y el potencial de cálculo de estas máquinas modernas, necesitamos una nueva disciplina de programación concurrente. En esta disciplina, diferentes tareas de cómputo se ejecutan en diferentes núcleos simultáneamente, lo que puede mejorar significativamente la eficiencia general de la ejecución. Sin embargo, a pesar de sus beneficios en eficiencia, la programación concurrente sigue siendo difícil, no solo por los nuevos lenguajes de programación que requiere, sino también por las complicadas interacciones entre las tareas de ejecución que conlleva.

# ANTECEDENTES: THREADS

Históricamente, una computadora solía tener un solo procesador, a menudo denominado maquina monoprocesador. En estas máquinas, las tareas de cómputo (normalmente procesos) se multiplexan en este único procesador mediante el cambio de tarea. Aunque este paradigma de programación nos da la ilusión de que estas tareas se ejecutan concurrentemente, de hecho, se ejecutan de manera secuencial.

a figure
to describe the multi-process model 

Hoy en día, la mayoría de las computadoras son maquinas multicore. Para programar tales computadoras, podríamos continuar utilizando el modelo de programación multiproceso, con un proceso ejecutándose en un núcleo. Sin embargo, este modelo tiene algunas desventajas que lo hacen menos ideal. En primer lugar, el proceso es una abstracción relativamente pesada con semánticas complicadas, así como altos costos de cambio de tarea. Además, resulta engorroso comunicarse entre procesos. Los mecanismos de sincronización o comunicación están limitados (por ejemplo, las tuberías (pipes) solo se pueden usar entre procesos padre e hijo) o son pesados (por ejemplo, los sockets normalmente atraviesan toda la pila de redes).

**CONCURRENCIA CON THREADS**

Un thread (hilo o hebra) es un nuevo modelo para ejecutar múltiples tareas simultáneamente en múltiples núcleos. A diferencia de los procesos, los hilos son ligeros en naturaleza, por lo tanto, es económico crearlos. Además, los hilos en un mismo proceso comparten el mismo espacio de direcciones, por lo que los datos (globales) y el código del programa pueden compartirse directamente entre todos los hilos, lo que lleva a un modelo de comunicación mucho más simple y económico.

POSIX soporta las siguientes especificaciones de las funciones thread, entre otras:

#include <pthread.h>

Int pthread\_create(pthread\_t \*thd. Const pthread\_attr\_t \*attr, void \*(\*start) (void \*), void \*arg;

Void pthread\_exit(void \*arg);

Int pthread\_join(void \*arg);

Int pthread\_detach(pthread\_t thread);

Pthread\_t pthread\_self(void);

Estas funciones gestionan todo el ciclo de vida de un hilo:

* **pthread\_create**: crea un nuevo hilo y comienza a ejecutar la función especificada.
* **pthread\_exit**: termina el hilo actual.
* **pthread\_join**: espera a que el hilo especificado por ‘*thd*’ termine y opcionalmente puede recuperar el valor de salida del hilo con el puntero ‘*ptr*’.
* **pthread\_detach**: separa el hilo. Una vez separado, el hilo reclamará automáticamente los recursos cuando termine.
* **pthread\_self**: obtiene el identificador de hilo del hilo que realiza la llamada actualmente.

# OBJETIVOS

El objetivo de este laboratorio es que el estudiante comprenda el concepto de hilos, unidad fundamental de utilización de la CPU que conforma los fundamentos de los sistemas informáticos multitarea, aprenderá cómo realizar programación concurrente y como programar con la abstracción básica de la programación concurrente: Los hilos (threads).

Para realizar los siguientes ejercicios, descarga el siguiente archivo:

wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1l5209YOadbKru3HBRcbRHr1IoCOXyvSq' -O Lab4.zip

# Ejercicio 1. (40pts)

**El archivo square.c proporcionado calcula la sumatoria de los cuadrados de los números enteros desde 0 hasta n:**

Formula de sumatoria que representa la suma de los cuadrados de números enteros desde 0 hasta n:

∑n𝑖=0 𝑖2=0∗0+1∗1+2∗2+…+𝑛∗𝑛

**También se proporciona el archivo squarethreads.c el cual realiza el mismo calculo, pero utilizando hilos.**

**Ejecutar cada programa, medir sus tiempos de ejecución y explicar lo sucedido.**

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Al ejecutar los dos códigos, observamos una diferencia en la velocidad de ejecución: el primer código es más rápido que el segundo. Esto se debe a que, para este tipo de cálculos simples, es más eficiente trabajar con un solo hilo. La creación, administración y sincronización de hilos adicionales en el segundo código genera una sobrecarga significativa en el sistema, lo que consume más recursos que la operación principal en sí misma.

# Ejercicio 2. (50pts)

**Lea el archivo pi.c proporcionado e investiga cómo implementar hilos en dicho programa, ejecutar cada programa utilizando 10000 por parámetro, medir sus tiempos de ejecución y explicar lo sucedido. (incluir captura del código implementado y de los tiempos).**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

En la siguiente captura de código se puede observar cómo se implementaron los hilos en el programa. En este caso, también se nota un aumento en el tiempo de ejecución del segundo código en comparación con el primero. Aunque la operación es más compleja que en el ejemplo anterior, no lo es lo suficiente como para que el uso de hilos proporcione una mejora significativa en el rendimiento. A partir de esto, se pueden destacar las siguientes observaciones:

* El uso de hilos es útil en tareas intensivas en cálculos que pueden paralelizarse eficientemente, como algoritmos que requieran múltiples iteraciones pesadas o cuando se puede aprovechar el hardware multinúcleo.
* En estos casos particulares, donde cada tarea es simple y rápida, el costo de la gestión de hilos supera el beneficio de paralelizar el trabajo.

**Texto

Descripción generada automáticamente**

# Entrega (10pts)

Documente (screenshots) cada una de las acciones antes señaladas. Es de exclusiva responsabilidad del estudiante respetar el formato de entrega de informe de esta guía **(debajo de cada enunciado su screenshot en donde aparezca de forma clara las sentencias utilizadas)**. El formato de entrega debe ser en PDF, y el nombre del archivo debe contener su nombre y apellido (**Laboratorio\_4\_Nombre\_Apellido**). Todas las actividades deben ser entregadas (subidas) a la plataforma digital en las fechas establecidas, por cada hora de atraso, se descontará 1 pto.