PREGUNTAS DE CHEQUEO – CAPÍTULO 4
SECCIÓN 4.1
4.1. ¿Cuántos hilos maneja un proceso tradicional (heavyweight)?
RPTA:
En la computación tradicional, un proceso tiene un solo hilo de ejecución, y por eso se le llama un proceso "peso pesado". Este único hilo incluye registros, contador de programas, pila, código, datos y archivos abiertos.

4.2. Mencione al menos tres beneficios de la programación multihilo.
RPTA:
Tres beneficios de la programación multihilo son:

- **Mayor capacidad de respuesta** Si un hilo se bloquea en una operación de entrada/salida (I/O), los otros hilos pueden seguir ejecutándose, mejorando la respuesta de la aplicación.
- **Compartir recursos** Es más rápido compartir información entre hilos que entre procesos, ya que los hilos dentro de un proceso comparten secciones de código, datos y archivos abiertos.
- **Economía** Crear un hilo requiere menos recursos en comparación con la creación de un proceso completo. Esto se debe a que no es necesario reservar todas las zonas de memoria. Además, las aplicaciones que utilizan hilos son más escalables que las que utilizan múltiples procesos.

SECCIÓN 4.2
4.3. (Verdadero / Falso) La concurrencia solamente es posible con paralelismo.
RPTA:
La afirmación "La concurrencia solamente es posible con paralelismo" es **falsa**.

El video explica que **puede haber concurrencia sin necesidad de paralelismo**. En un sistema con un solo núcleo, los procesos pueden ejecutarse de forma concurrente, donde el sistema operativo cambia entre las tareas, dando la apariencia de que progresan simultáneamente. Sin embargo, el paralelismo implica que varios hilos se ejecutan simultáneamente en diferentes núcleos de procesamiento, lo que mejora la concurrencia al permitir que los hilos avancen más rápidamente.

4.4. (Verdadero / Falso) La ley de Amdahl se refiere al efecto desproporcionado de la porción serial de un programa.
RPTA:
La afirmación "La ley de Amdahl se refiere al efecto desproporcionado de la porción serial de un programa" es **Verdadera**.

La ley de Amdahl muestra el efecto desproporcionado que tiene sobre la aceleración del sistema el hecho de que los procesos tengan partes que se tienen que ejecutar en serie. No todas las tareas que vaya a ejecutar un proceso o un hilo se pueden hacer en paralelo, sino que a veces hay tareas que son seguidas y esas se ejecutan en un solo hilo.

La fórmula de la ley de Amdahl indica que la aceleración que se puede obtener en un proceso es menor o igual que una expresión donde `n` es el número de núcleos y `s` es el porcentaje de la tarea que es serial.

Por ejemplo, si el 5% de las tareas de un proceso deben ejecutarse de manera serial, la aceleración se reduce dramáticamente. Con dos núcleos, la velocidad se duplica, pero con cuatro núcleos, ya no es cuatro veces más rápida, sino alrededor de 3.8 veces. Con 10 núcleos, la aceleración es de siete, no 10 veces, y con 16 núcleos, la velocidad de ejecución se multiplica por nueve. Si el 50% de la tarea es serial, no se puede obtener una aceleración mayor que dos.

4.5. Mencione al menos tres retos de la programación en sistemas multicore.
RPTA:
Tres retos de la programación en sistemas multinúcleo son:

- **Identificar tareas que se puedan dividir y ejecutar de manera concurrente**. Este es el primer reto a la hora de programar aplicaciones en un sistema multinúcleo.
- **Equilibrio en la paralelización**. Hay tareas que no se benefician mucho de la paralelización, por lo que no vale la pena el esfuerzo de paralelizar partes de la aplicación que no aporten mucho al desempeño.
- **División de datos**. Al paralelizar, es necesario dividir los datos, ya que cada hilo puede emplear datos diferentes. Es importante considerar qué porción de los datos necesita cada hilo para evitar problemas de comunicación y lentitud.
- **Pruebas y depuración**. Hacer pruebas y depurar una aplicación multihilo es más difícil que hacerlo con una aplicación de un solo hilo, debido a la posibilidad de múltiples caminos de ejecución y problemas de sincronización.

4.6. ¿Cuáles son los dos tipos generales de paralelismo?
RPTA:
Los dos tipos generales de paralelismo son **paralelismo de datos** y **paralelismo de tareas**.

- **Paralelismo de datos:** Implica particionar los datos y hacer que cada núcleo realice una tarea determinada sobre su porción de datos. Por ejemplo, un algoritmo de ordenamiento donde cada núcleo ordena una parte de los datos y luego se reúnen los resultados.
- **Paralelismo de tareas:** Implica realizar varias tareas sobre los mismos datos. Por ejemplo, calcular la media, la desviación estándar, la moda, el valor máximo y el valor mínimo de un

vector simultáneamente en diferentes núcleos.

Estos dos métodos no son excluyentes y se puede emplear un híbrido entre ambos.

SECCIÓN 4.3
4.7. Enuncie tres maneras comunes de mapear hilos de usuario a hilos de kernel.
RPTA:
Tres maneras comunes de mapear hilos de usuario a hilos de kernel son:

- **Muchos a uno** Múltiples hilos de usuario se mapean a un solo hilo de kernel. Si uno de los hilos de usuario se bloquea, bloquea el progreso de todos los hilos. Este esquema era común en sistemas que no eran *multicore*.
- **Uno a uno** Por cada hilo de usuario, hay un hilo de kernel correspondiente. El número de hilos que se pueden crear puede estar restringido por la capacidad del kernel. Windows y Linux emplean este esquema, donde cada hilo en el espacio de usuario se mapea inmediatamente a un hilo en el espacio del kernel.
- **Muchos a muchos** Múltiples hilos de usuario pueden ser mapeados a múltiples hilos de kernel simultáneamente. Un hilo de usuario podría estar corriendo en varios hilos de kernel al mismo tiempo, sin las restricciones del modelo muchos a uno. La programación de este esquema es muy complicada y no es muy común. Aunque no es común, se puede lograr en Windows instalando un paquete llamado *threat fiber*.

SECCIÓN 4.4
4.8. ¿Cuáles son los dos métodos para implementar una librería de hilos?
RPTA:
El video explica que existen **dos métodos principales para implementar una librería de hilos**: **Hilos de usuario** e **Hilos de kernel**.

- **Hilos de usuario**: Son los hilos que se programan en las aplicaciones, utilizando librerías como las Pthreads para sistemas Unix/Linux/macOS, la librería propia de hilos de Java, o la API de hilos de Windows. Los hilos de usuario son manipulados directamente por el programador al crear un programa.
- **Hilos de kernel**: Son gestionados por el núcleo del sistema operativo y se encargan de ejecutar las tareas de los hilos de usuario. Existe una comunicación entre los hilos de usuario y los hilos de kernel.

Además, el video describe **diferentes maneras de mapear los hilos de usuario a los hilos de kernel**:

- **Muchos a uno**: Múltiples hilos de usuario se mapean a un solo hilo de kernel.
- **Uno a uno**: Por cada hilo de usuario, hay un hilo de kernel correspondiente.

- **Muchos a muchos**: Múltiples hilos de usuario pueden ser mapeados a múltiples hilos de kernel simultáneamente.
- **Modelo de dos niveles**: Un hilo de usuario podría estar corriendo en varios hilos de kernel, pero a la vez se puede configurar que un hilo en particular esté mapeado a un solo hilo de kernel.

4.9. Enuncie las tres principales librerías de hilos que se emplean en la actualidad.
RPTA:
Las tres principales librerías de hilos que se emplean en la actualidad, según el video, son:

- Para sistemas Unix, Linux o macOS, se emplea una librería llamada **Pthreads**.
- **Java** tiene su propia librería de hilos.
- **Windows** también maneja su propia API de hilos.

SECCIÓN 4.5
4.10. Enuncie al menos dos técnicas para soporte de threading implícito.
RPTA:
Dos técnicas para el soporte de **threading implícito** que se mencionan en el video son:

- **Thread pools de Java**: Permite definir un "pool" de hilos que las distintas tareas dentro del proceso pueden utilizar. Cuando una tarea termina, el hilo se libera de nuevo al pool. El programador no tiene que preocuparse por crear, borrar o gestionar los hilos.
- **Paralelismo "fork-join" en Unix**: Se le da al compilador la instrucción de que parta la tarea en dos procesos y que luego una los resultados.
- **OpenMP**: Permite marcar secciones del código como paralelizadas mediante directivas de compilador especiales llamadas "pragma". El compilador se encarga del resto, haciendo que el uso de hilos sea transparente para el programador.
- **Grand Central Dispatch en macOS e iOS**: Similar a OpenMP, este esquema permite marcar partes del código como paralelizadas y el sistema operativo se encarga del resto.
- **Intel Thread Building Blocks**: De manera similar a los anteriores, permite marcar ciertas partes del código como paralelizadas, delegando al sistema operativo la gestión de los hilos.

El **threading implícito** permite que el programador identifique tareas que puedan ejecutarse simultáneamente y las marque en el código, sin necesidad de gestionar los hilos directamente. El compilador o el sistema operativo se encargan de crear, gestionar y eliminar los hilos, simplificando el proceso de programación.