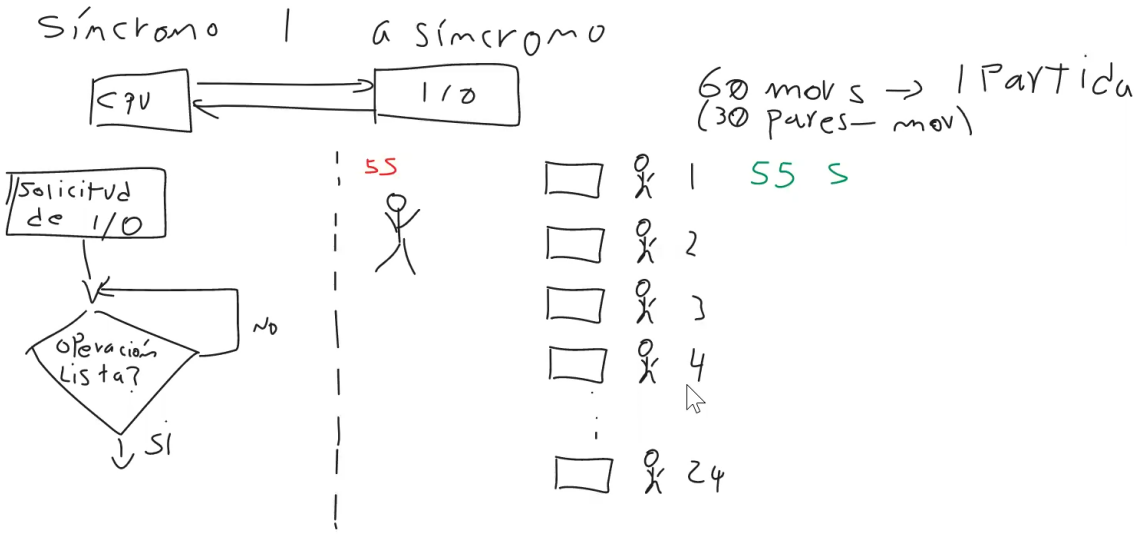
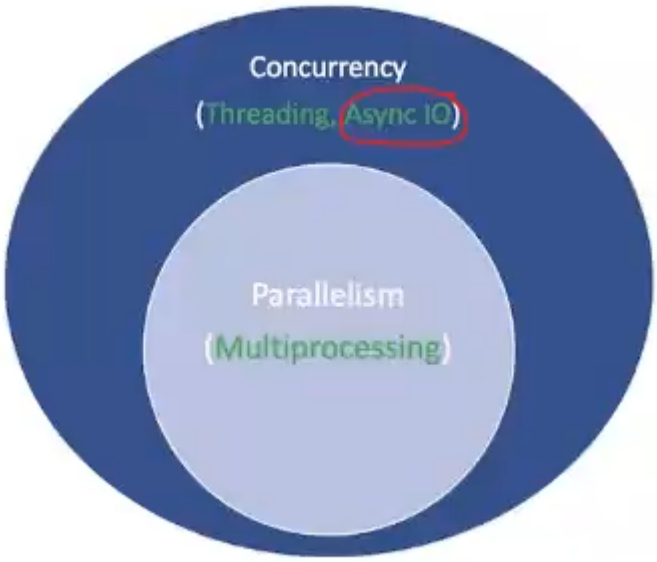
PC4

Síncrono vs. asíncrono





Preguntas

Responder los siguientes incisos teóricos de manera concisa y con sus propias palabras:

1. ¿Qué es el kernel?
    - El kernel es el núcleo o corazón de un sistema operativo. Es la parte central responsable de administrar los recursos del sistema, brindar servicios a las aplicaciones y facilitar la comunicación entre el hardware y el software.
2. ¿Qué son las llamadas al sistema y cuál es su función?
    - Las llamadas al sistema son interfaces proporcionadas por el sistema operativo para que las aplicaciones puedan solicitar servicios o realizar operaciones específicas, como leer o escribir en un archivo, crear un proceso o realizar operaciones de red. Estas llamadas permiten que las aplicaciones interactúen con el sistema operativo y utilicen sus funcionalidades.
3. ¿Las llamadas al sistema son dependientes del sistema operativo?
    - Sí, las llamadas al sistema son dependientes del sistema operativo. Cada sistema operativo implementa su propio conjunto de llamadas al sistema con sus propias convenciones y parámetros. Por lo tanto, las llamadas al sistema pueden variar entre sistemas operativos diferentes.
4. ¿Qué es un hilo?
    - Un hilo (thread) es una unidad básica de ejecución dentro de un proceso. Representa una secuencia de instrucciones que pueden ejecutarse de manera independiente, compartiendo recursos y espacio de memoria con otros hilos dentro del mismo proceso.
5. ¿Qué contiene un hilo?
    - Un hilo contiene su propia secuencia de ejecución, incluyendo el contador de programa, la pila de ejecución y los registros. Además, comparte recursos como la memoria, los archivos abiertos y otros recursos del proceso al que pertenece.
6. ¿Qué comparten los hilos?
    - Los hilos comparten recursos del proceso al que pertenecen, como la memoria y los archivos abiertos. También pueden compartir variables y datos en memoria, lo que les permite comunicarse y cooperar entre sí.
7. ¿Qué es un proceso?

- Un proceso es una instancia en ejecución de un programa. Representa un entorno de ejecución independiente con su propio espacio de memoria, recursos asignados y una o más hilos de ejecución. Un proceso puede contener uno o varios hilos.

8. ¿Qué recursos se asignan al crear un proceso?
    - Al crear un proceso, se asignan recursos como espacio de memoria, identificadores de archivos abiertos, variables y estructuras de datos. También se asigna un identificador único para el proceso, que se utiliza para administrarlo y realizar operaciones relacionadas con él.
9. ¿Cuál es la diferencia entre concurrencia y paralelismo?
    - La concurrencia se refiere a la capacidad de ejecutar múltiples tareas de manera simultánea, independientemente de si se ejecutan en paralelo físicamente o no. Implica el progreso simultáneo de varias tareas, que pueden compartir recursos pero no necesariamente ejecutarse al mismo tiempo.
    - El paralelismo, por otro lado, implica la ejecución real simultánea de múltiples tareas en diferentes unidades de procesamiento. En este caso, las tareas se ejecutan verdaderamente al mismo tiempo y se pueden beneficiar de los sistemas con múltiples núcleos o procesadores.
10. ¿Qué es multihilo?
    - Multihilo se refiere al uso de múltiples hilos dentro de un proceso. Permite que un programa divida su carga de trabajo en tareas más pequeñas y las ejecute concurrentemente para aprovechar mejor los recursos del sistema y mejorar la capacidad de respuesta.
11. ¿Qué es multiproceso?
    - Multiproceso se refiere al uso de múltiples procesos simultáneamente. Cada proceso tiene su propio espacio de memoria y se ejecuta como una entidad independiente. Los procesos pueden comunicarse entre sí, pero tienen su propio entorno de ejecución aislado. Esto se puede aprovechar para lograr la concurrencia y el paralelismo en sistemas con múltiples núcleos o procesadores.

Verdadero o falso

Responda verdadero o falso según corresponda y justifique su respuesta en caso sea falsa.

1. Puedes moverte libremente entre cualquiera de los estados de un proceso.
    - Falso. Los procesos en un sistema operativo pueden moverse entre diferentes estados, como en ejecución, listo, bloqueado o terminado, pero no es posible moverse libremente entre ellos. El cambio de estado está determinado por el planificador del sistema operativo y las interrupciones.
2. Cada programa tiene asociado un solo proceso del sistema.
    - Falso. Un programa puede tener asociado más de un proceso del sistema. Por ejemplo, en sistemas operativos multitarea, pueden ejecutarse múltiples instancias de un mismo programa como procesos independientes.
3. No existe un real paralelismo en la computadora.
    - Falso. En las computadoras modernas con múltiples núcleos o procesadores, es posible lograr un paralelismo real. Esto significa que múltiples tareas pueden ejecutarse simultáneamente en diferentes unidades de procesamiento, lo que mejora la eficiencia y el rendimiento del sistema.
4. Es posible crear una infinita cantidad de procesos dentro de una computadora.
    - Falso. La creación de procesos está limitada por los recursos del sistema, como la memoria y el tiempo de CPU. En la práctica, hay un límite en la cantidad de procesos que se pueden crear dentro de una computadora, y ese límite varía según el sistema operativo y la configuración del hardware.
5. Solo existen dos tipos de interfaces de usuario: CLI y GUI.
    - Falso. Además de las interfaces de usuario de línea de comandos (CLI) y las interfaces gráficas de usuario (GUI), existen otros tipos de interfaces de usuario, como las interfaces de usuario basadas en voz, las interfaces táctiles y las interfaces basadas en gestos, entre otras.
6. Existe un único sistema de archivos para todos los sistemas operativos.
    - Falso. Cada sistema operativo tiene su propio sistema de archivos. Aunque puede haber sistemas de archivos compatibles o comunes entre diferentes sistemas operativos, no existe un único sistema de archivos que sea utilizado por todos los sistemas operativos.
7. Cada programa en ejecución tiene su propio contador de programa (Program Counter).

- Verdadero. Cada programa en ejecución tiene su propio contador de programa (program counter), que indica la dirección de la próxima instrucción a ejecutar. El contador de programa se actualiza a medida que se ejecutan las instrucciones del programa.

8. El sistema operativo es independiente del hardware, es decir, que puede ser instalado en cualquier máquina.
    - Falso. El sistema operativo está estrechamente relacionado con el hardware subyacente y está diseñado para funcionar con un conjunto específico de componentes y dispositivos. No todos los sistemas operativos pueden instalarse en cualquier máquina, ya que se requiere compatibilidad con los controladores y la arquitectura del hardware.
9. Las aplicaciones orientadas al usuario no interactúan directamente con el hardware del sistema.
    - Verdadero. Las aplicaciones orientadas al usuario generalmente interactúan con el sistema operativo a través de interfaces de programación de aplicaciones (API) proporcionadas por el sistema operativo. Estas API actúan como una capa de abstracción que permite a las aplicaciones comunicarse con el hardware del sistema a través del sistema operativo, sin tener que interactuar directamente con el hardware.

Async

La programación asíncrona, o "async" por su abreviatura, es un enfoque de programación que se utiliza para realizar tareas de manera concurrente y sin bloquear el flujo de ejecución principal de un programa. En lugar de esperar a que una tarea se complete antes de pasar a la siguiente, se permite que el programa siga ejecutando otras tareas mientras se espera a que una operación asincrónica se complete en segundo plano.

El uso de async es especialmente útil en situaciones donde se realizan operaciones que pueden llevar tiempo, como la lectura o escritura de archivos, solicitudes a una base de datos o peticiones a través de una red. En lugar de detener por completo la ejecución del programa mientras se espera a que estas operaciones se completen, se puede utilizar async para continuar ejecutando otras tareas y aprovechar mejor los recursos del sistema.

En la programación asíncrona, las tareas se dividen en bloques más pequeños y se ejecutan de manera independiente. Se utilizan mecanismos como promesas,

callbacks o async/await para gestionar la asincronía y coordinar la ejecución de las tareas. Esto permite que múltiples operaciones se realicen simultáneamente, mejorando así el rendimiento y la capacidad de respuesta del programa.

Al utilizar async, es posible realizar operaciones en paralelo, aprovechando los recursos de hardware disponibles, como múltiples núcleos o procesadores. Esto puede llevar a un mejor rendimiento y a una ejecución más eficiente de las tareas.

En resumen, async es un enfoque de programación que permite realizar tareas de forma concurrente y sin bloquear el flujo principal de un programa, lo que mejora el rendimiento y la capacidad de respuesta del mismo al aprovechar los recursos de hardware y permitir la ejecución simultánea de múltiples tareas

Multithreading

El multithreading, o "multihilo", es una técnica de programación que permite la ejecución simultánea de múltiples hilos dentro de un programa. Un hilo es una secuencia de instrucciones que puede ejecutarse de forma independiente y concurrente con otros hilos en un proceso.

El uso de multithreading tiene como objetivo mejorar el rendimiento y la eficiencia de los programas al aprovechar los recursos del sistema de manera más efectiva. Al dividir una tarea en hilos más pequeños, se puede ejecutar cada hilo en paralelo, utilizando múltiples núcleos o procesadores disponibles en el hardware. Esto permite que las tareas se realicen más rápidamente y mejora la capacidad de respuesta de la aplicación.

Los hilos comparten recursos y memoria dentro de un proceso, lo que facilita la comunicación y la cooperación entre ellos. Sin embargo, también es necesario tomar precauciones para evitar problemas de concurrencia, como las condiciones de carrera o la falta de sincronización en el acceso a recursos compartidos.

El multithreading se utiliza en situaciones donde hay tareas que pueden ejecutarse de manera independiente y paralela, como la gestión de la interfaz de usuario en aplicaciones gráficas, la ejecución de operaciones de red en segundo plano o el procesamiento paralelo de grandes conjuntos de datos.

En resumen, el multithreading es una técnica de programación que permite la ejecución simultánea de múltiples hilos dentro de un programa. Al dividir una tarea en hilos más pequeños y ejecutarlos en paralelo, se mejora el rendimiento y la eficiencia de la aplicación al aprovechar los recursos del sistema de manera más efectiva. Sin embargo, también se deben tomar precauciones para evitar problemas de concurrencia.

Multiprocessing

Multiprocessing, o "multiproceso", es una técnica de programación que permite la ejecución simultánea de múltiples procesos independientes dentro de un sistema. Un proceso es una instancia en ejecución de un programa, y cada proceso tiene su propio espacio de memoria y recursos asignados.

El uso de multiprocessing tiene como objetivo mejorar el rendimiento y la capacidad de respuesta de los programas al aprovechar los recursos del sistema de manera más efectiva. A diferencia del multithreading, donde los hilos comparten recursos dentro de un proceso, en el multiprocessing cada proceso tiene su propio entorno de ejecución aislado. Esto permite una mayor escalabilidad y puede aprovechar mejor los sistemas con múltiples núcleos o procesadores.

Los procesos pueden comunicarse entre sí a través de mecanismos de intercambio de información, como tuberías, colas o memoria compartida. Esto facilita la colaboración y la coordinación entre los procesos.

El multiprocessing se utiliza en situaciones donde las tareas se pueden dividir en procesos independientes que pueden ejecutarse en paralelo. Es especialmente útil para tareas intensivas en CPU, procesamiento de datos masivos o aplicaciones que se benefician de una mayor distribución y paralelismo.

Sin embargo, el multiprocessing también conlleva cierta complejidad adicional, ya que los procesos requieren una mayor cantidad de recursos y existe una sobrecarga asociada con la creación y gestión de múltiples procesos.

En resumen, el multiprocessing es una técnica de programación que permite la ejecución simultánea de múltiples procesos independientes dentro de un sistema. Permite aprovechar los recursos del sistema de manera más efectiva, mejorar el rendimiento y la capacidad de respuesta de los programas. Se utiliza en situaciones donde las tareas pueden dividirse en procesos independientes que se ejecutan en paralelo, pero también introduce cierta complejidad adicional.