UNIVERSIDAD NACIONAL DE INGENIERÍA
DIRECCIÓN DE ÁREA DE CONOCIMIENTO DE TECNOLOGIAS DE INFORMACIÓN Y COMUNICACIÓN
INGENIERÍA EN COMPUTACIÓN

Descripciones Generales			
Asignatura:	***Sistemas Distribuidos***	***Semestre Académico:***	***II***
Año Lectivo:	***2025***		
Docente:	***Ing. Araceli Torres López***		

Clase práctica	***5***	***Unidad III:***	***Procesos y Procesadores***
Tema de Clase práctica	• ***Hilos*** • ***Modelos de Sistemas Operativos***		
Objetivos	***Comprender los conceptos fundamentales de gestión de procesos e hilos, diferenciando sus características, funcionamiento y el contexto en el que ocurre el pseudoparalelismo en los sistemas operativos.*** ***Analizar las estrategias de asignación y replicación de procesos en sistemas distribuidos, identificando los mecanismos que permiten la tolerancia a fallas, el equilibrio de carga y la optimización del uso de recursos.*** ***Reconocer las distintas arquitecturas de sistemas operativos (monolítica, microkernel e híbrida) y explicar las funciones y responsabilidades del núcleo en cada enfoque.***		

UNIVERSIDAD NACIONAL DE INGENIERÍA
DIRECCIÓN DE ÁREA DE CONOCIMIENTO DE TECNOLOGIAS DE INFORMACIÓN Y COMUNICACIÓN
INGENIERÍA EN COMPUTACIÓN

Actividades de desarrollo

1. **Formar grupos de 4 a 5 personas**.
2. **Haga lectura reflexiva y analítica sobre el contenido del archivo de la Unidad III-Hilos (Threads) en los Sistemas Operativos Modelos de Sistemas Operativos, para contestar a las interrogantes planteadas en la actividad de aprendizaje.**

Actividades de Aprendizaje

Nombre: Darling Lilieth Jiménez Rosales **Carnet:**2018-0087U

Conteste de manera argumentativa:

i. ¿Cuál es la característica principal de un hilo (thread) que lo diferencia de un proceso?

La característica principal que distingue a un hilo de un proceso es que los hilos comparten el mismo espacio de memoria y recursos, mientras que los procesos son completamente independientes y aislados entre sí.

Un proceso es como un programa en ejecución con su propio conjunto de recursos: memoria, archivos abiertos y estado del procesador. Si abres dos veces el mismo programa, creas dos procesos separados, cada uno en su propia "burbuja". En cambio, un hilo es una unidad de ejecución más pequeña dentro de un proceso. Múltiples hilos pueden coexistir en un solo proceso, compartiendo su código, datos y archivos.

Esta diferencia es crucial. Como los hilos comparten memoria, la comunicación entre ellos es mucho más rápida y eficiente que la comunicación entre procesos (que requiere mecanismos más complejos como tuberías o memoria compartida gestionada por el sistema operativo). Sin embargo, esta misma característica introduce riesgos: un error en un hilo puede corromper los datos de todo el proceso y afectar a los demás hilos. En resumen, la principal diferencia radica en el aislamiento de recursos: los procesos están aislados, mientras que los hilos son colaborativos.

ii. ¿En qué contexto se produce el pseudoparalelismo?

El pseudoparalelismo ocurre en sistemas que tienen un único procesador (CPU) pero que necesitan ejecutar múltiples tareas (procesos o hilos) de forma concurrente. En este contexto, el sistema operativo crea la ilusión de que las tareas se están ejecutando simultáneamente, cuando en realidad no es así.

El "truco" consiste en que el planificador (scheduler) del sistema operativo asigna pequeños intervalos de tiempo de CPU a cada tarea de manera muy rápida. Cambia de una tarea a otra a una velocidad tan alta que para el usuario parece que todo sucede al mismo tiempo. Por ejemplo, puedes estar escribiendo en un documento, escuchando música y descargando un archivo a la vez. Aunque la CPU solo puede hacer una de esas cosas en un instante preciso, el cambio es tan veloz que se percibe como una ejecución paralela.

En contraste, el paralelismo real solo se logra en sistemas con múltiples núcleos o procesadores, donde varias tareas pueden, literalmente, ejecutarse en el mismo instante de tiempo. Por lo tanto, el pseudoparalelismo es una técnica de gestión de la concurrencia en sistemas monoprocesador.

iii. ¿Cómo se gestionan los hilos en un sistema operativo distribuido, a diferencia de un sistema centralizado?

La gestión de hilos cambia radicalmente entre un sistema centralizado y uno distribuido debido a la naturaleza de la arquitectura subyacente.

En un sistema centralizado, todos los hilos de un proceso se ejecutan en una única máquina. El núcleo del sistema operativo local tiene control total sobre ellos: los crea, los destruye y los planifica utilizando la CPU y la memoria de esa misma máquina. La comunicación entre hilos es extremadamente rápida porque acceden a la misma memoria compartida.

En un sistema operativo distribuido, los hilos de un mismo proceso pueden estar ejecutándose en diferentes máquinas conectadas por una red. Esto introduce una complejidad enorme. La gestión ya no depende de un solo núcleo, sino de una coordinación entre múltiples sistemas operativos. La comunicación entre hilos que están en nodos distintos es mucho más lenta, ya que debe viajar por la red. Además, surgen nuevos desafíos como la sincronización global (asegurar que los hilos en diferentes máquinas accedan a recursos compartidos de forma ordenada) y la tolerancia a fallos (¿qué pasa si una de las máquinas falla?).

En esencia, en un sistema centralizado la gestión es local y eficiente, mientras que en uno distribuido es coordinada, compleja y dependiente de la red.

iv. ¿Cuál de las siguientes funciones NO se encuentra típicamente en el núcleo de una arquitectura microkernel?

En una arquitectura de microkernel (micronúcleo), la función que típicamente NO se encuentra en el núcleo es la gestión del sistema de archivos (file system).

La filosofía del microkernel es mantener el núcleo lo más pequeño y esencial posible, incluyendo únicamente las funciones más críticas, como:

- La gestión básica de procesos y hilos (creación y planificación).
- La gestión de la memoria (asignación de espacios de memoria).
- La comunicación entre procesos (IPC - Inter-Process Communication).

Casi todo lo demás, como los controladores de dispositivos (drivers), los protocolos de red y, notablemente, el sistema de archivos, se implementa como servicios que se ejecutan en el espacio de usuario, fuera del núcleo. Esto contrasta con los núcleos monolíticos (como el de Linux tradicional), donde todas estas funciones residen dentro del núcleo. La ventaja del microkernel es que es más modular, seguro y estable (un fallo en el sistema de archivos no colapsará todo el sistema), pero a costa de un rendimiento potencialmente menor debido a la sobrecarga de comunicación entre los servicios y el núcleo.

v. En la gestión de procesos en sistemas distribuidos, ¿qué problema se busca resolver con un protocolo de "verificación y bloqueo" al usar estaciones de trabajo inactivas?

El protocolo de "verificación y bloqueo" busca resolver el problema de la reclamación inoportuna de recursos. Este problema surge cuando un proceso de un usuario es asignado a una estación de trabajo que estaba inactiva, pero de repente el dueño de esa estación regresa y comienza a usarla.

Imaginemos que el sistema asigna una tarea pesada a tu computadora porque no la estabas usando. Justo en ese momento, regresas y abres un programa de diseño gráfico. El sistema necesita una manera de manejar este conflicto. El protocolo funciona así:

Verificación: El sistema detecta que la estación de trabajo ha vuelto a estar activa (por ejemplo, al moverse el ratón o pulsar una tecla).

Bloqueo: En lugar de simplemente matar el proceso invitado, el protocolo puede "bloquear" o notificar al proceso original (o al gestor del sistema) que el recurso ya no está disponible.

Acción: A partir de ahí, se toma una decisión: o bien se migra el proceso invitado a otra estación de trabajo inactiva, o se le da un tiempo para que termine si está a punto de hacerlo, o en el peor de los casos, se termina.

El objetivo es garantizar que el propietario de la estación de trabajo siempre tenga prioridad y no sufra una degradación del rendimiento, al mismo tiempo que se gestiona de forma ordenada el destino de los procesos "invitados".

vi. ¿Qué característica define a un algoritmo de asignación de procesadores de tipo "distribuido"?

La característica que define a un algoritmo de asignación de procesadores de tipo "distribuido" es que la toma de decisiones se realiza de forma cooperativa entre las diferentes máquinas (nodos) del sistema, sin que exista una entidad central que lo controle todo.

En un enfoque centralizado, una sola máquina (el "coordinador") es responsable de monitorear la carga de todos los nodos y decidir dónde enviar cada nuevo proceso. Esto es simple, pero crea un cuello de botella y un punto único de fallo.

En un algoritmo distribuido, cada nodo tiene su propia parte de la lógica de asignación. Los nodos se comunican entre sí para compartir información sobre su carga de trabajo actual. Por ejemplo, un nodo sobrecargado puede "preguntar" a sus vecinos si tienen capacidad disponible para aceptar un nuevo proceso. La decisión se toma localmente, basándose en la información (posiblemente incompleta o desactualizada) que cada nodo tiene sobre el estado del resto del sistema. Esto los hace más escalables y robustos frente a fallos.

vii. ¿Cuál es la principal diferencia entre una estrategia de asignación de procesadores migratoria y una no migratoria?

La principal diferencia radica en la capacidad de mover un proceso que ya está en ejecución.

Una estrategia no migratoria es más simple: una vez que un proceso es asignado a una máquina, se ejecuta en esa máquina hasta que termina. La decisión se toma solo al inicio. Si esa máquina se sobrecarga más tarde, el proceso no se moverá. Es como elegir una fila en el supermercado: una vez que estás en ella, te quedas ahí hasta el final.

Una estrategia migratoria, en cambio, permite mover un proceso de una máquina a otra durante su ejecución. Esto es mucho más complejo, ya que implica "congelar" el proceso, empaquetar todo su estado (memoria, registros, etc.), transferirlo por la red a la nueva máquina y reanudarlo allí sin que se pierda información. La gran ventaja es que permite un balanceo de carga dinámico: si un nodo se sobrecarga, algunos de sus procesos pueden ser migrados a nodos menos ocupados para optimizar el rendimiento global del sistema.

En resumen, la diferencia clave es la flexibilidad: las estrategias migratorias son dinámicas y adaptativas, mientras que las no migratorias son estáticas.

viii. ¿Qué es la replicación de procesos en el contexto de la tolerancia a fallas?

La replicación de procesos es una técnica fundamental para lograr la tolerancia a fallas en sistemas distribuidos. Consiste en crear y mantener múltiples copias idénticas (réplicas) de un mismo proceso y ejecutarlas en diferentes máquinas del sistema. ♡

El objetivo es simple: si una de las máquinas que aloja una réplica falla (por ejemplo, se apaga o se desconecta de la red), las demás réplicas pueden continuar ejecutándose sin interrupción. De esta manera, el servicio que ofrece el proceso sigue estando disponible para los usuarios, ocultando el fallo de un componente individual.

Para que esto funcione, es crucial mantener la consistencia entre las réplicas. Es decir, todas deben recibir las mismas entradas en el mismo orden para que sus estados internos se mantengan sincronizados. Gestionar esta consistencia es uno de los mayores desafíos de la replicación.

ix. En el modelo de replicación pasiva (Activo-Pasivo), ¿cuál es el rol de las réplicas?

En el modelo de replicación pasiva, también conocido como primario-respaldo (primary-backup), el rol de las réplicas (las pasivas) es actuar como copias de seguridad inactivas que están listas para tomar el control si el proceso principal (el activo) falla.
El funcionamiento es el siguiente:

- Solo una réplica, la activa o primaria, procesa todas las solicitudes de los clientes.
- Después de procesar una solicitud, la réplica activa actualiza su estado y luego envía esa actualización a todas las réplicas pasivas.
- Las réplicas pasivas no hacen nada más que recibir estas actualizaciones y mantener su estado sincronizado con el del primario. No interactúan con los clientes.
- Si el primario falla, un mecanismo de detección de fallos lo identifica y una de las réplicas pasivas es promovida para convertirse en el nuevo primario. A partir de ese momento, ella comenzará a procesar las solicitudes.

El rol de las réplicas pasivas es, por tanto, de espera y respaldo, garantizando una rápida recuperación ante un fallo del componente principal.

x. ¿Cuál es el objetivo principal de una arquitectura de sistema operativo híbrido?

El objetivo principal de una arquitectura de sistema operativo híbrido es combinar las ventajas de las arquitecturas monolítica y de microkernel, mientras se minimizan sus respectivas desventajas.

De la arquitectura monolítica, busca heredar el alto rendimiento. Al mantener ciertos componentes críticos y de uso frecuente (como el sistema de archivos o los drivers gráficos) dentro del espacio del núcleo, se evitan los costosos cambios de contexto entre el modo usuario y el modo núcleo, logrando una comunicación más rápida.

De la arquitectura de microkernel, busca adoptar la modularidad, estabilidad y seguridad. Al mover otros servicios menos críticos al espacio de usuario, se logra que un fallo en uno de ellos no comprometa a todo el sistema. Además, permite una mayor flexibilidad para añadir o quitar funcionalidades.

Sistemas operativos modernos como Windows y macOS son ejemplos de arquitecturas híbridas. No son puramente monolíticos ni puramente microkernels, sino que implementan una solución pragmática que busca el equilibrio perfecto entre velocidad, robustez y flexibilidad para el uso diario.