1. Planificación de Procesos y Algoritmo de Planificación:

Planificación de Procesos: Decide qué proceso se ejecutará en la CPU en un momento dado. Garantiza asignación eficiente de recursos y rendimiento optimo del sistema.

Algoritmo de Planificación: Conjunto de reglas y políticas que determinan el orden en el que los procesos se ejecutan en la CPU.

Algoritmo: Toman decisiones sobre cuándo y por cuanto tiempo se asigna la CPU a cada proceso en la cola de procesos listos para la ejecución. Garantizan equidad, eficiencia y capacidad de respuesta.

2. Principales Objetivos de los Algoritmos de Planificación:

Manejar el uso de recursos optimizando

Orientado al sistema operativo: rendimiento, equidad, equilibrio

Orientado al usuario: prioridades, previsibilidad, tiempo de respuesta, tiempo de ejecución

Maximizar la utilización de la CPU: Buscan mantener la CPU ocupada tanto como sea posible para aprovechar al máximo su capacidad de procesamiento.

Maximizar la capacidad de respuesta del sistema: Se esfuerzan por responder rápidamente a las solicitudes de los usuarios y los procesos interactivos para proporcionar una experiencia fluida.

Minimizar el tiempo de espera de los procesos: Intentan reducir el tiempo que los procesos pasan en la cola de espera, lo que mejora la eficiencia y la velocidad de ejecución.

Evitar la inanición: Garantizan que todos los procesos tengan la oportunidad de ejecutarse y no sean relegados permanentemente debido a la prioridad o la llegada de otros procesos.

3. Dificultades en la Planificación de Procesos:

Lograr todos los objetivos simultáneamente puede ser complicado, ya que a veces los objetivos pueden entrar en conflicto. Por ejemplo, dar prioridad a la capacidad de respuesta puede llevar a una menor utilización de la CPU.

La variabilidad en la llegada de procesos y sus requisitos de tiempo de CPU hace que la planificación sea desafiante. Algunos procesos pueden ser impredecibles en términos de su duración o comportamiento.

La toma de decisiones sobre qué proceso debe ejecutarse a continuación puede ser subjetiva y depende de la política de planificación elegida.

4. Clasificación de Algoritmos de Planificación:

Planificador de Corto Plazo (CPU Scheduling):

- Se encarga de decidir qué proceso en ejecución se ejecutará en la CPU en un momento dado.
- Su objetivo principal es optimizar la asignación de la CPU y la eficiencia del sistema.

Planificador de Mediano Alcance (Admisión):

- Gestiona la admisión de nuevos procesos en la memoria principal desde el almacenamiento secundario.
- Considera la capacidad de memoria y puede aplicar políticas de reemplazo.
- Su función es equilibrar la eficiencia y la capacidad del sistema.

Planificador de Largo Plazo (Job Scheduling):

- Se encarga de seleccionar qué procesos se cargarán en la memoria desde el almacenamiento secundario.
- Decisión fundamental para la gestión de la carga de trabajo en el sistema.
- Puede utilizar políticas como la planificación basada en lotes.

5. Algoritmo FIFO (First Come First Served):

El algoritmo FIFO asigna la CPU al proceso que llegó primero a la cola de procesos listos.

Es simple y fácil de implementar, pero puede llevar a tiempos de espera largos, ya que no tiene en cuenta la duración de los procesos.

6. Algoritmo Round Robin:

1 - Cada proceso se ejecuta durante un tiempo fijo conocido como "quantum".

2 - El proceso se mueve al final de la cola de procesos listos, lo que permite a otros procesos obtener tiempo de CPU.

3 - Es justo y evita la inanición, pero puede tener un alto costo de cambio de contexto si el quantum es demasiado corto.

7. Importancia de Definir con Cuidado la Duración del Quantum:

La duración del quantum en el algoritmo Round Robin es un parámetro crítico.

Si el quantum es demasiado largo, los procesos pueden volverse no responsivos, lo que afecta la capacidad de respuesta del sistema.

Si el quantum es demasiado corto, se produce una sobrecarga significativa debido al cambio frecuente de contexto, lo que también puede reducir la eficiencia general del sistema.

La elección del quantum depende de la naturaleza de los procesos y los objetivos del sistema operativo.

8. Algoritmo de Planificación por Prioridades:

En el algoritmo de planificación por prioridades, cada proceso tiene asignada una prioridad.

La CPU se asigna al proceso con la prioridad más alta en ese momento.

Este enfoque permite dar preferencia a ciertos tipos de procesos (por ejemplo, procesos interactivos) sobre otros (procesos en segundo plano).

Sin embargo, si no se maneja adecuadamente, los procesos de baja prioridad pueden experimentar inanición si los procesos de alta prioridad siempre están disponibles.

9. Evitar que Procesos de Baja Prioridad no se Ejecuten:

Para evitar la inanición de procesos de baja prioridad, se pueden implementar políticas de envejecimiento. Estas políticas aumentan gradualmente la prioridad de los procesos que han estado esperando en la cola de prioridad baja durante mucho tiempo, lo que les da la oportunidad de ejecutarse.

También se puede establecer un límite mínimo de prioridad que un proceso puede tener para garantizar que los procesos no se degraden indefinidamente.

10. Algoritmo de Planificación de Colas Múltiples:

- En el algoritmo de planificación de colas múltiples, la cola de procesos se divide en varias colas, cada una con una prioridad diferente.

- Cada cola puede utilizar un algoritmo de planificación diferente. Por ejemplo, la cola de alta prioridad puede usar Round Robin, mientras que la cola de baja prioridad puede usar FIFO.

- Este enfoque permite aplicar políticas de planificación adecuadas a diferentes tipos de procesos y equilibrar la capacidad de respuesta y la eficiencia.

11. Algoritmo del Primer Trabajo Más Corto (SPN):

- El algoritmo SPN selecciona el proceso más corto para ejecutar primero.

- Este enfoque minimiza el tiempo de espera promedio de todos los procesos, ya que los procesos más cortos se ejecutan primero.

- Sin embargo, requiere conocimiento previo de la duración de los procesos, lo que puede ser difícil de obtener en la práctica.

- SPN es adecuado para sistemas donde se conocen los tiempos de ejecución con precisión y se prioriza minimizar el tiempo de espera.

12. Algoritmo del Tiempo Remanente más Corto Primero (SRT):

- El algoritmo SRT es similar a SPN, pero permite la preempción. Si llega un proceso más corto que el que está en ejecución, se realiza un cambio de contexto y se ejecuta el proceso más corto.

- SRT es altamente eficiente en términos de tiempo de respuesta, ya que se adapta dinámicamente a las llegadas de nuevos procesos.

- Sin embargo, puede generar sobrecarga por cambio de contexto debido a la frecuente preempción.

13. Recomendación y Condiciones para SPN y SRT:

- Estos algoritmos son recomendables en sistemas de tiempo compartido donde los usuarios valoran tiempos de respuesta rápidos.

- Las condiciones ideales son tener conocimiento preciso de los tiempos de ejecución de los procesos y la presencia de procesos interactivos de corta duración.

- SPN y SRT no son adecuados para sistemas donde la variabilidad en los tiempos de ejecución es alta o donde no se pueden prever con precisión.

14. Estimación del Tiempo de Ejecución de Procesos:

- La estimación del tiempo de ejecución de los procesos es crucial para algoritmos como SPN y SRT.

- Se pueden utilizar técnicas de estimación, como el promedio ponderado de los tiempos anteriores de ejecución de un proceso o modelos estadísticos para predecir cuánto tiempo tomará un proceso.

- Estas estimaciones se actualizan continuamente a medida que el proceso se ejecuta y se ajustan según sea necesario.

15. Planificación con Más de un Procesador:

- En sistemas con múltiples procesadores (SMP), la planificación se complica por la presencia de varias CPUs.

- Se deben utilizar algoritmos de asignación de procesos para determinar qué procesos se ejecutarán en qué CPU para aprovechar el paralelismo de manera eficiente.

- Esto permite una mayor capacidad de procesamiento y rendimiento, especialmente en sistemas con cargas de trabajo intensivas.

16. Planificación de Hilos con Más de un Procesador:

- La planificación de hilos en sistemas multiprocesador se centra en asignar hilos de manera eficiente a las CPU disponibles.

- Se deben evitar situaciones en las que demasiados hilos compitan por un número limitado de CPU, lo que podría generar una sobrecarga por cambio de contexto.

- Los sistemas operativos y las bibliotecas de programación proporcionan herramientas para controlar la afinidad de los hilos con CPU específicas y administrar el paralelismo de manera óptima.

17. Asignación de Hilos a Procesadores

Procesador dedicado: Cada procesador es estático, sus hilos permanecen dentro del mismo procesador constantemente, aprovecha al máximo la cache del procesador.

Planificación dinámica: Los procesadores son compartidos globalmente, los hilos pueden variar entre procesadores disponibles, los procesadores eligen que proceso ejecutar primero.

Compartir la carga: Los procesadores son compartidos globalmente, los hilos del proceso pueden variar entre procesadores disponibles, mayor balanceo de carga

Asignación en banda: Los procesadores son compartidos globalmente, todos los hilos del proceso pueden usar todos los procesadores disponibles, mayor concurrencia dentro de un proceso

18. Conceptos.

 Multi-Programación: Capacidad de poder gestionar muchos procesos (en diferentes estados) al mismo tiempo.

 Multi-Tarea: "Sensación'' de que se están ejecutando varios Procesos 'al mismo tiempo', aún cuando existe disponible un único Procesador.

 Multi-Threading: Es una extensión de la Multi-Tarea aplicada a los Hilos de los Procesos por lo que un Proceso podría realizar varias cosas 'al mismo tiempo'.

 Multi-Procesamiento: Computadora que posee más de un Procesador por lo que puede ejecutar verdaderamente más de un Proceso (o Hilo de un Proceso) al mismo tiempo.

 (Procesador) Multi-Núcleo: Procesador que tiene la capacidad de ejecutar verdaderamente más de un Hilo del mismo Proceso al mismo tiempo.

Esclavo/Maestro vs Peer en Sistemas Operativos:

Esclavo/Maestro (Master/Slave): En este modelo, un nodo o componente es designado como el "maestro" y toma decisiones y controla a los "esclavos". Es útil en sistemas donde se necesita una jerarquía y centralización de control, como bases de datos distribuidas o sistemas de control industrial.

Peer (Pares): En este modelo, todos los nodos se consideran iguales y pueden comunicarse directamente. No hay un nodo central de control. Es común en sistemas distribuidos descentralizados, como redes peer-to-peer, donde cada nodo tiene igual capacidad y responsabilidad.

Estática vs Dinámica en Sistemas Operativos:

Estática: En sistemas operativos, "estática" se refiere a propiedades que no cambian o cambian muy lentamente con el tiempo. Por ejemplo, el análisis estático de un sistema se enfoca en sus propiedades en un momento dado, sin considerar cómo evolucionan con el tiempo. En programación, el análisis estático de código implica revisar el código sin ejecutarlo.

Dinámica: En contraste, "dinámica" implica cambio y movimiento con el tiempo. En sistemas operativos, el análisis dinámico observa cómo las propiedades de un sistema cambian mientras está en funcionamiento. En programación, el análisis dinámico implica ejecutar el programa y rastrear su comportamiento en tiempo real.