**2.2 Modelo Multi-hilo**

2.2.1 Uso de Hilos

* **Hilos en sistemas operativos tradicionales:**

En sistemas operativos tradicionales, cada proceso tiene un espacio de direcciones y un solo hilo de control.

En situaciones donde se necesite ejecutar múltiples actividades de manera cuasi-paralela dentro del mismo espacio de direcciones, se utilizan hilos.

* **Razones para usar hilos:**

Simplificación del modelo de programación al descomponer una aplicación en varios hilos secuenciales.

Facilidad y rapidez en la creación y destrucción de hilos, ya que son más ligeros que los procesos.

Mejora del rendimiento al permitir la ejecución de actividades intensivas en CPU y operaciones de E/S de manera simultánea.

Utilidad en sistemas con varias CPUs, donde se puede lograr verdadero paralelismo.

* **Aplicación de hilos:**

Procesador de palabras: Permite realizar tareas como formateo en segundo plano mientras se interactúa con el usuario.

Hoja de cálculo electrónica: Facilita el cálculo de elementos mientras se realizan modificaciones adicionales.

Servidor web: Permite manejar múltiples solicitudes concurrentes de manera eficiente, como el acceso a páginas en caché y la gestión de operaciones de disco sin bloqueo.

* **Beneficios de los hilos**

Posibilitan el concepto de procesos secuenciales con llamadas al sistema con bloqueo, manteniendo el paralelismo y mejorando el rendimiento.

Permiten manejar eficientemente aplicaciones que procesan grandes cantidades de datos mediante la ejecución simultánea de operaciones de entrada, salida y procesamiento.

* Es importante que las llamadas al sistema bloqueen solo al hilo que las realizó, no a todo el proceso, para evitar ineficiencias en la utilización de la CPU.

2.2.2 El modelo clásico de hilo

* **Modelo de procesos:**

Un proceso agrupa recursos relacionados, como espacio de direcciones, archivos abiertos, procesos hijos, alarmas pendientes, entre otros.

Cada proceso tiene al menos un hilo de ejecución, que incluye un contador de programa, registros de variables de trabajo y una pila.

* **Añadiendo hilos al modelo de procesos:**

Los hilos permiten ejecuciones múltiples y casi independientes dentro del mismo proceso.

Los hilos comparten el mismo espacio de direcciones y otros recursos dentro del proceso, lo que los hace "procesos ligeros" o "procesos en miniatura".

* **Comparación entre procesos e hilos:**

Los hilos tienen propiedades similares a los procesos, pero con algunas diferencias clave, como el compartimiento de variables globales y la capacidad de acceder a la misma memoria.

Mientras que los procesos pueden ser de diferentes usuarios y hostiles entre sí, los hilos de un mismo proceso cooperan entre sí.

* **Estados de los hilos:**

Al igual que los procesos, los hilos pueden estar en estados como ejecución, bloqueado, listo o terminado.

Cada hilo tiene su propia pila que contiene el historial de ejecución de los procedimientos llamados.

* **Creación y terminación de hilos:**

Los hilos pueden ser creados y terminados mediante llamadas a procedimientos de biblioteca, como thread\_create y thread\_exit.

Algunos sistemas permiten que un hilo espere a que otro termine mediante la llamada thread\_join.

* **Gestión de la CPU:**

Los hilos deben cooperar y permitir que otros hilos tengan acceso a la CPU utilizando llamadas como thread\_yield.

* **Complicaciones en el modelo de programación:**

Introducen problemas adicionales, como la gestión de hilos en procesos hijos, el cierre de archivos compartidos entre hilos y la asignación de memoria redundante.

* **Necesidad de diseño cuidadoso:**

Es esencial pensar y diseñar cuidadosamente programas con multi-hilo para evitar problemas de concurrencia y asegurar un funcionamiento correcto.

2.2.3 Hilos en POSIX

* **Estándar IEEE 1003.1c y Pthreads:**

El estándar IEEE 1003.1c define el paquete de hilos conocido como Pthreads, que proporciona una interfaz portátil para la programación multi-hilo en sistemas UNIX.

Pthreads es ampliamente aceptado en la mayoría de los sistemas UNIX y proporciona más de 60 llamadas a funciones para la gestión de hilos.

* **Propiedades de los hilos Pthreads:**

Cada hilo Pthreads tiene las siguientes propiedades:

* + - Identificador único.
    - Conjunto de registros, incluyendo el contador de programa.
    - Conjunto de atributos, que se almacenan en una estructura y pueden incluir el tamaño de la pila, parámetros de planificación y otros elementos necesarios para utilizar el hilo.

**5 Planificación de la CPU**

5.1 Conceptos básicos

**Multiprogramación y Ciclo de Ráfagas de CPU y E/S:**

* + La multiprogramación permite que varios procesos se ejecuten de manera concurrente para maximizar el uso de la CPU.
  + Los procesos alternan entre ráfagas de CPU (ejecución en la CPU) y ráfagas de E/S (espera por operaciones de entrada/salida).
  + Este ciclo se repite continuamente durante la ejecución de un proceso.

**Planificador de la CPU:**

* + Es responsable de seleccionar qué proceso en la cola de procesos listos se ejecutará a continuación en la CPU.
  + La cola de procesos listos puede implementarse de diversas formas, como FIFO, prioritaria, entre otras.

**Planificación Apropiativa vs Sin Desalojo:**

* + La planificación apropiativa permite que un proceso sea desalojado de la CPU antes de que termine su ráfaga de CPU.
  + En la planificación sin desalojo, un proceso se ejecuta hasta completar su ráfaga de CPU o hasta que se bloquee.

**Despachador:**

* + Es el módulo encargado de realizar el cambio de contexto y transferir el control de la CPU entre procesos.
  + Debe ser eficiente para minimizar la latencia de cambio de contexto.

**Manejo de Interrupciones:**

* + Se necesitan mecanismos para gestionar las interrupciones que puedan ocurrir mientras se ejecutan los procesos.
  + Las secciones críticas del código que manejan interrupciones suelen desactivarlas temporalmente para evitar problemas de concurrencia.

5.2 Criterios de planificación

**Utilización de la CPU:**

* + Se busca mantener la CPU ocupada tanto como sea posible, dentro de un rango saludable (generalmente entre el 40% y el 90%).
  + Una CPU muy ocupada indica una alta eficiencia en la ejecución de procesos.

**Tasa de procesamiento:**

* + Mide la cantidad de procesos completados por unidad de tiempo.
  + Es una medida directa del rendimiento del sistema en términos de productividad.

**Tiempo de ejecución:**

* + Es el tiempo total que tarda un proceso desde que se ordena su ejecución hasta que se completa.
  + Incluye períodos de espera en la cola de procesos preparados, ejecución en la CPU y operaciones de E/S.

**Tiempo de espera:**

* + Es el tiempo total que un proceso pasa esperando en la cola de procesos preparados antes de ser ejecutado.
  + Se busca minimizar este tiempo para mejorar la eficiencia y la capacidad de respuesta del sistema.

**Tiempo de respuesta:**

* + Es el tiempo transcurrido desde que se envía una solicitud hasta que se produce la primera respuesta del proceso.
  + Es crucial en sistemas interactivos para garantizar una respuesta rápida a las solicitudes del usuario.

El objetivo es optimizar la utilización de la CPU y la tasa de procesamiento, mientras se minimizan el tiempo de ejecución, el tiempo de espera y el tiempo de respuesta. En muchos casos, se busca optimizar algún tipo de valor promedio, pero también puede ser importante optimizar los valores máximo y mínimo, especialmente en situaciones donde se requiera un rendimiento predecible y consistente.

5.3 Algoritmos de planificación

**Planificación FCFS (Primero en llegar, primero en ser servido):**

* + Asigna la CPU al proceso que llega primero a la cola de procesos preparados.
  + Implementado fácilmente con una cola FIFO.
  + Tiende a tener un tiempo medio de espera largo, especialmente si las ráfagas de CPU de los procesos son muy variables.
  + Puede causar el efecto convoy, donde procesos más cortos tienen que esperar detrás de procesos largos.
  + Tiempo medio de espera:

**Planificación SJF (Selección del trabajo más corto):**

* + Asigna la CPU al proceso con la siguiente ráfaga de CPU más corta.
  + Puede ser no apropiativo (no desaloja al proceso actualmente en ejecución) o apropiativo (desaloja el proceso actual si llega uno con una ráfaga de CPU más corta).
  + Suele proporcionar el tiempo medio de espera mínimo para un conjunto de procesos dado.
  + La dificultad radica en conocer la duración de la siguiente ráfaga de CPU.
  + Puede ser aproximado mediante la predicción de la siguiente ráfaga de CPU utilizando un promedio exponencial.
  + Tiempo medio de espera:
  + Promedio exponencial para predicción de la siguiente ráfaga de CPU: , donde es la predicción de la próxima ráfaga de CPU, es la duración de la n-ésima ráfaga de CPU, es el valor predicho anteriormente, y es un factor de suavizado entre 0 y 1.

**Planificación por prioridades:**

* + Asigna la CPU al proceso con la prioridad más alta.
  + Los procesos con la misma prioridad se planifican en orden FCFS.
  + Puede ser apropiativo (desaloja al proceso actual si llega uno con una prioridad más alta) o cooperativo (coloca el nuevo proceso al principio de la cola de procesos preparados).
  + Puede llevar al bloqueo indefinido o a la muerte por inanición de procesos de baja prioridad, que pueden solucionarse mediante mecanismos de envejecimiento.
  + Tiempo medio de espera:
  + Mecanismo de envejecimiento: Disminuir gradualmente la prioridad de los procesos en espera durante un período de tiempo especificado.

Video:

**1. Introducción a la planificación de procesos:**

- Definición de proceso y sus estados.

- Importancia de la planificación para el funcionamiento eficiente del sistema operativo.

**2. Objetivos de la planificación de procesos:**

- Equidad en la ejecución de tareas.

- Maximización de la utilización de la CPU.

- Aumento de la productividad.

- Minimización del tiempo de espera, retorno y respuesta.

**3. Algoritmos de planificación:**

- FIFO: Primero en entrar, primero en salir.

- SJF: Trabajo más corto primero.

- Round Robin: Ejecución por turnos con límite de tiempo.

**4. Algoritmos SJF (No apropiativo y apropiativo):**

- Ejecución del proceso más corto primero.

- Comparación de ráfagas de CPU para determinar el próximo proceso.

- Cálculo de tiempos de espera y retorno.

**5. Planificación por prioridad:**

- Ejecución según la prioridad asignada a cada proceso.

- Comparación de prioridades y selección del próximo proceso.

- Cálculo de tiempos de espera y retorno.

**6. Round Robin:**

- Ejecución de procesos en bloques de tiempo fijo.

- Aplicación de FIFO y limitación de tiempo de ejecución por proceso.

- Cálculo de tiempos de espera y retorno.

**7. Selección del algoritmo más eficiente:**

- Comparación de tiempos promedio de espera y retorno para elegir el algoritmo adecuado.