ARQUITECTURA Y SISTEMAS OPERATIVOS

Unidad 2 – PROCESOS

a. Procesos: Definiciónes - Modelo de Procesos. Procesos e hilos.
b. Planificación de procesos: FCFS - SJF - Round-Robin - por Prioridad.
c. Comunicación entre procesos: Procesos Concurrentes. Condiciones de competencia. Secciones críticas. Exclusión mutua. Semáforos.

CONCEPTOS BASICOS

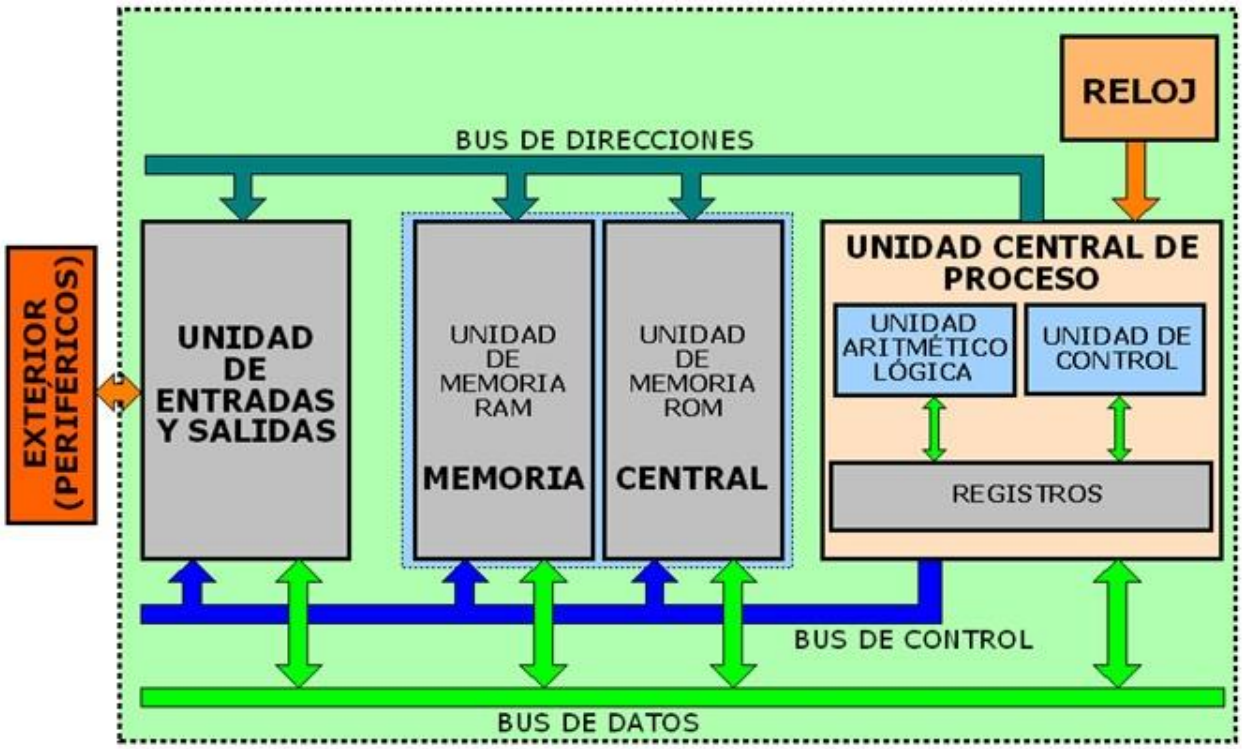
A continuación, se desglosan conceptos fundamentales a partir de lecturas de internet e IA, a modo de glosario, para ofrecer una visión general y simplificada de la unidad, contenidos que deben ser profundizados con la bibliografía de la asignatura.

Procesos: Definiciones - Modelo de Procesos

Definición: un proceso es básicamente un programa en ejecución que incluye el contador de programa, los registros del procesador y las variables. Es la instancia de un programa en ejecución y es la unidad básica de ejecución en un sistema operativo.

Modelo de Procesos: El modelo de procesos en sistemas operativos describe las distintas fases por las que pasa un proceso durante su ciclo de vida. Las fases típicas incluyen la creación, la ejecución, la espera (bloqueo) y la terminación. El sistema operativo gestiona múltiples procesos manteniendo su estado y administrando su alternancia en la CPU mediante mecanismos de planificación.

Recordemos el esquema básico



Procesos e Hilos

Un proceso es un programa en ejecución y necesita de ciertos recursos como: CPU, memoria, archivos, dispositivos E/S para llevar a cabo su tarea. Estos recursos se asignan cuando se crea o ejecuta un Proceso. Es decir que los procesos, son entidades independientes que tienen su propio contexto de ejecución completo (memoria, contadores de programa, etc.). Cada proceso se ejecuta en su propio espacio de dirección.

Todo sistema tiene una colección de procesos: los procesos del S.O ejecutan código del sistema y los procesos de usuario ejecutan código de usuario. Todos estos procesos pueden ejecutarse en forma concurrente.

Un programa es una entidad pasiva almacenada en disco (archivo ejecutable), mientras que un proceso es una entidad activa.

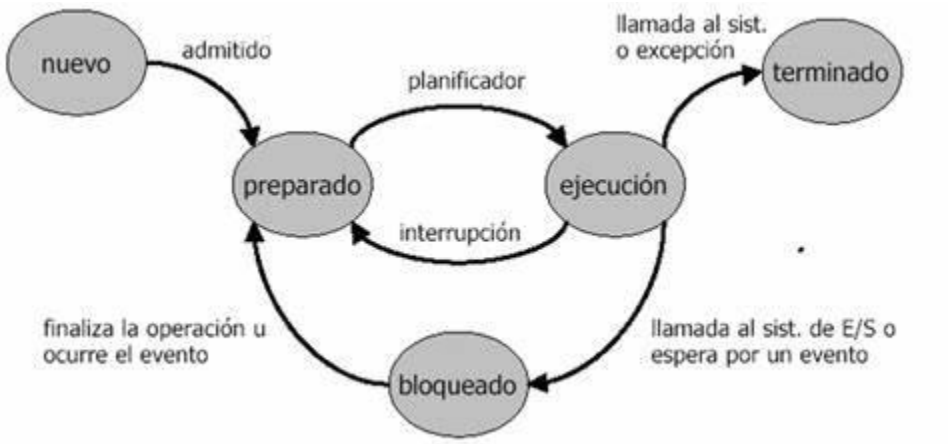
Un programa se convierte en proceso cuando se carga en memoria un archivo ejecutable.

Existen dos técnicas habituales para cargar archivos ejecutables:
- Doble clic al ícono del archivo ejecutable
- vía línea de comandos digitando el nombre del archivo

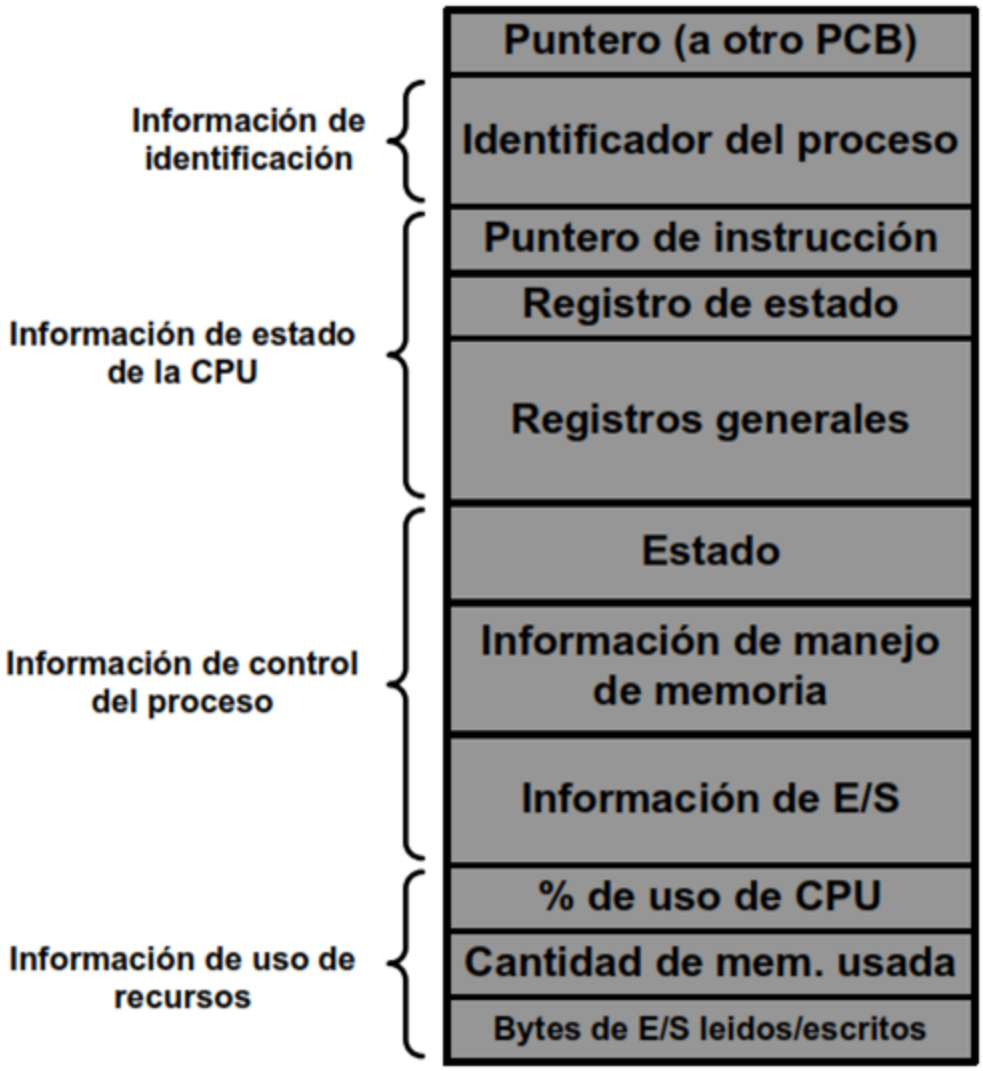
Un proceso incluye:
-Código del programa (código de programa de usuario)
-Contador de programa (actividad actual, se apoya con el contenido de los registros del procesador)
-Pila (datos temp como parámetros de funciones, direcciones de retorno y variables locales)
-Sección de datos (variables globales)

-Head (memoria asignada en tiempo de ejecución)

Estados de un proceso:
Conforme se ejecuta un proceso cambia su estado, el cual puede ser:
- nuevo: el proceso se está creando
- en ejecución: se están ejecutando sus instrucciones
- en espera o bloqueado: está esperando que ocurra algún evento (ej. E/S)
- preparado o listo: está esperando que le asignen la CPU
- terminado: ha terminado su ejecución
Puede haber muchos procesos preparados y en espera.



Bloque de Control de Procesos BCP (PCB)
Cada proceso se representa en el SO mediante un PCB. Un PCB contiene elementos como:
-Estado del proceso: (ejecución, espera, etc.)
-Contador de programa: (apunta a sig. instrucción a ser ejecutada)
-Registros de la CPU: (guarda info de registros de CPU, indicadores de estado y contador del prog cuando hay una interrupción así, luego puede continuar ejecutándose)
-Información de planificación de CPU: (prioridad del proceso, punteros a colas de planificación, etc.)
-Información de gestión de memoria: (información de memoria asignada al proceso)
-Información contable: (tiempo real de CPU empleado, lim time asignado, núm. proceso, etc.)
-Información de estado de E/S: (lista de disp. E/S asignados al proceso, etc.)



Hilos:
Son entidades dentro de un proceso que pueden ejecutarse de manera concurrente. Comparten el espacio de memoria del proceso y sus recursos, pero cada hilo tiene su propia pila y contador de programa. Los hilos permiten realizar múltiples tareas dentro del mismo proceso de manera eficiente.
Threads.- (Hilos). - un proceso puede ser un solo hilo de ejecución (hilo de instrucciones).
Con los hilos existe la posibilidad de tener varios contadores de programa por proceso lo cual implica que varias instrucciones pueden ejecutar a la vez. Los contadores de programa de cada proceso se almacenan en los PBC de cada proceso.

Administración de Procesos

Un **Administrador de Procesos** es un componente del sistema operativo que se encarga de la creación, gestión, y terminación de procesos. Es responsable de coordinar todos los aspectos relacionados con los procesos en el sistema, lo que incluye la asignación de recursos (como CPU y memoria), el mantenimiento del estado de los procesos, la planificación de su ejecución, y la gestión de la interacción entre ellos. A continuación, se describen las funciones principales de un administrador de procesos:

1. Creación y Destrucción de Procesos

El administrador de procesos maneja la creación de nuevos procesos, a menudo como resultado de una solicitud de un usuario o de otro proceso. Esto incluye la asignación de los recursos necesarios y la configuración inicial. Igualmente, se encarga de la terminación de los procesos, liberando cualquier recurso que hubieran estado utilizando.

2. Planificación de Procesos

Una de las funciones más críticas del administrador de procesos es la planificación de cuándo y cómo los procesos acceden al procesador. Esto se realiza mediante **algoritmos de planificación** que pueden incluir métodos como Round-Robin, Shortest Job First (SJF), Prioridad, entre otros. La elección del algoritmo de planificación afecta directamente la eficiencia y la equidad del sistema operativo.

3. Sincronización y Comunicación entre Procesos

El administrador de procesos también gestiona la sincronización y la comunicación entre los procesos concurrentes. Esto implica controlar el acceso a recursos compartidos para evitar conflictos y condiciones de carrera, utilizando mecanismos como semáforos, bloqueos (locks) y variables de condición.

4. Manejo de Estados de Procesos

Los procesos pueden encontrarse en varios estados, como listo, en ejecución, bloqueado o terminado. El administrador de procesos mantiene el seguimiento de estos estados para gestionar adecuadamente la ejecución de cada proceso, moviéndolos entre diferentes colas de espera según sea necesario para su procesamiento.

5. Gestión de Recursos

El administrador de procesos también está involucrado en la asignación y liberación de otros recursos además de la CPU, como la memoria y los dispositivos de entrada/salida. Esto incluye la gestión del espacio de direcciones de memoria de cada proceso y la asignación de tiempo de acceso a dispositivos de E/S.

6. Interfaz con Otros Componentes del Sistema

Finalmente, el administrador de procesos trabaja en conjunto con otros componentes del sistema operativo, como el sistema de archivos, el gestor de memoria, y el sistema de entrada/salida, para asegurar que los procesos puedan realizar sus tareas efectivamente.

El administrador de procesos es esencial para el funcionamiento eficiente y organizado de un sistema operativo, asegurando que los recursos del sistema se utilicen de manera justa y efectiva mientras se maximiza el rendimiento general del sistema.

Planificación de Procesos

El término **Planificación de Procesos** hace referencia a un conjunto de políticas y mecanismos del SO que gobiernan el **orden** en que se ejecutan los procesos.

Los mecanismos de planificación de la CPU son la base de los S.O multiprogramados.
Por medio de la conmutación de la CPU entre distintos procesos el S.O hace que el computador sea más productivo.
Por ello es importante conocer los conceptos básicos sobre la planificación de la CPU y los algoritmos utilizados para este fin, así como seleccionar el mejor algoritmo para un sistema en particular.

El **planificador de procesos** es una parte específica del administrador de procesos que decide qué proceso se ejecuta a continuación y por cuánto tiempo en el CPU. Su objetivo principal es optimizar el uso del CPU y mejorar el rendimiento del sistema operativo al asignar recursos de manera efectiva entre los procesos.

El planificador de procesos se centra específicamente en la asignación eficiente del CPU entre los procesos disponibles, tomando decisiones sobre la ejecución y priorización de los procesos en función de políticas de planificación específicas. Las políticas de planificación se Implementan para lograr objetivos específicos, como maximizar la utilización del CPU, minimizar el tiempo de respuesta o garantizar la equidad en el acceso al CPU entre procesos.

Mientras el **administrador de procesos** abarca una gama más amplia de responsabilidades relacionadas con la gestión y coordinación general de los procesos, el **planificador de procesos** se enfoca en la asignación efectiva del CPU para optimizar el rendimiento del sistema operativo.

Los objetivos principales de la planificación de procesos son:
-Equidad: todos los procesos deben poder ejecutarse sin esperas largas
-Eficacia: mantener ocupada la CPU un 100% del tiempo
-Tiempo de respuesta: minimizar al máximo el tiempo de respuesta al usuario
-Rendimiento: maximizar el número de tareas o procesos ejecutados por segundo, minuto u hora.

ALGORITMOS DE PLANIFICACIÓN DE PROCESOS

FCFS (First-Come, First-Served): Es el algoritmo de planificación más simple donde el primer proceso que llega es el primero en ser atendido. No es preemptivo y puede llevar a la condición conocida como "anomalía de convoy".
En este algoritmo el tiempo de espera para que un proceso se ejecute es incierto y no mínimo.
Pudiendo así ejecutarse dentro de la CPU un proceso que consuma demasiado tiempo, atrasando a otros procesos y dejando la CPU sin trabajo por lapsos de tiempo.
En definitiva, este algoritmo hace que los procesos pequeños (en relación de ráfagas de CPU o tiempo de uso de CPU) esperen a que un proceso grande abandone la CPU. Esto se convierte en una gran desventaja.

SJF (Shortest Job First): Este algoritmo selecciona el proceso con el menor tiempo de ejecución próximo. Puede ser no preemptivo o preemptivo (a menudo llamado SRTF, Shortest Remaining Time First). Mejora considerablemente el tiempo de respuesta promedio.
Asigna la CPU al proceso cuya siguiente ráfaga de CPU es más corta.
Si dos procesos empatan se resuelve el empate por FCFS
Dos posibilidades:

No expropiativo (Cooperativo) - cuando se asigna la CPU a un proceso no se puede expropiar hasta que completa su ráfaga de CPU
Expropiativo (Apropiativo, **SRTF**) - si llega un proceso a la cola de listos con una ráfaga de CPU más corta que el tiempo que le queda al proceso en ejecución, se expropia.
SJF es óptimo - da el mínimo tiempo de espera medio para un conjunto de procesos dado, pero requiere conocer de antemano la duración de la siguiente ráfaga de CPU

Por Prioridad: Asigna a cada proceso una prioridad y planifica según la prioridad. Los procesos de alta prioridad se ejecutan primero. Puede ser preemptivo o no preemptivo.
Se asocia con cada proceso una prioridad (número entero).
La CPU se asigna al proceso con la prioridad más alta (consideramos número pequeño=prioridad alta)
Tenemos dos posibilidades:
-Expropiativo
-No expropiativo

SJF se puede ver como un algoritmo de planificación por prioridad en el que la prioridad es la duración predicha para la siguiente ráfaga de CPU.
Problema: Inanición (Bloqueo indefinido) - los procesos de más baja prioridad podrían no ejecutarse nunca.
Solución: Envejecimiento - conforme el tiempo pasa aumentar la prioridad de los procesos que esperan mucho en el sistema.

El término "**preemptivo**" en el contexto de los sistemas operativos se refiere a la capacidad de interrumpir o pausar un proceso que está en ejecución para asignar el tiempo de CPU a otro proceso. Este concepto es fundamental en el diseño de los algoritmos de planificación de procesos y se opone al método no preemptivo, donde un proceso no puede ser interrumpido hasta que finaliza su ejecución o hasta que realiza una operación que requiera esperar. Como ejemplo tenemos Round Robin (RR): Cada proceso recibe un pequeño intervalo de tiempo fijo (quantum), tras el cual el proceso es interrumpido y el siguiente proceso en la cola recibe CPU y Prioridades Preemptivas: Los procesos con mayor prioridad interrumpen la ejecución de procesos de menor prioridad.

Round-Robin: El algoritmo Round Robin es uno de los algoritmos de planificación de CPU más comunes y simples. Es particularmente útil en entornos donde se requiere equidad en la asignación de tiempo de CPU entre múltiples procesos. Es un algoritmo preemptivo que asigna un tiempo fijo (quantum) a cada proceso en un orden circular. Es especialmente eficiente en sistemas con muchos procesos.

Funcionamiento Básico:

-Cola de Listos (Ready Queue): Todos los procesos listos para ejecutarse se colocan en una cola circular llamada "cola de listos" o "ready queue".

-Asignación de Tiempo: Cada proceso en la cola de listos recibe un pequeño intervalo de tiempo de CPU llamado "quantum" o "quantum de tiempo".

-Ejecución por Turnos: Los procesos se ejecutan en turnos basados en el orden de llegada a la cola de listos. Cada proceso recibe una porción de tiempo de CPU igual al quantum.

-Interrupción por Quantum: Cuando un proceso ha consumido su quantum de tiempo, se suspende temporalmente y se coloca al final de la cola de listos, permitiendo que el siguiente proceso en la cola

tome su turno. El proceso suspendido conserva su estado para continuar desde donde se detuvo cuando vuelva a estar en ejecución.

-Ciclo Continuo: Este proceso de ejecución se repite continuamente, asignando tiempo de CPU a cada proceso en la cola de listos en un ciclo circular.

Características Clave:

Equidad: Debido a que cada proceso recibe un quantum de tiempo de CPU igual, el algoritmo Round Robin garantiza un trato equitativo entre los procesos. Ningún proceso puede acaparar el tiempo de CPU de manera indefinida.

Respuesta Rápida e implementación simple: Los procesos de alta prioridad pueden comenzar a ejecutarse rápidamente ya que están en la parte delantera de la cola de listos.

Implementación de políticas de planificación para lograr objetivos específicos, como maximizar la utilización del CPU, minimizar el tiempo de respuesta o garantizar la equidad en el acceso al CPU entre procesos.

Administrador y Planificador de procesos:

- El administrador de procesos es responsable de la gestión general de todos los procesos en el sistema, incluyendo su creación, destrucción y gestión de recursos.

- El planificador de procesos se centra específicamente en la asignación eficiente del CPU entre los procesos disponibles, tomando decisiones sobre la ejecución y priorización de los procesos en función de políticas de planificación específicas.

El administrador de procesos abarca una gama más amplia de responsabilidades relacionadas con la gestión y coordinación general de los procesos, mientras que el planificador de procesos se enfoca en la asignación efectiva del CPU para optimizar el rendimiento del sistema operativo.

Comunicación entre Procesos

A continuación, veremos un resumen de los conceptos relacionados con la comunicación entre procesos, centrándome en aspectos como procesos concurrentes, condiciones de competencia, secciones críticas, exclusión mutua y semáforos, con el objetivo de comprenderlos y facilitar su aplicación en simuladores de sistemas operativos.

Procesos Concurrentes

Los procesos concurrentes son aquellos que se ejecutan de forma simultánea o aparentemente simultánea en un sistema operativo. Esto significa que múltiples procesos están activos y comparten los recursos del sistema, como la CPU y la memoria, al mismo tiempo. Estos procesos pueden estar relacionados entre sí o pueden ser completamente independientes.

Condiciones de Competencia

Las condiciones de competencia, también conocidas como condiciones de carrera o condiciones de carrera crítica, son situaciones en las que el resultado de la ejecución de un programa depende del orden en que se ejecutan ciertas operaciones. Esto puede llevar a resultados inesperados o inconsistentes si no se gestionan adecuadamente.

Secciones Críticas

Una sección crítica es una parte de un programa o algoritmo en el que un proceso accede o modifica recursos compartidos que pueden ser utilizados por otros procesos concurrentes. Es importante gestionar adecuadamente las secciones críticas para evitar condiciones de competencia y garantizar la consistencia y la integridad de los datos.

Exclusión Mutua

La exclusión mutua es un principio fundamental en la programación concurrente que establece que, en un momento dado, solo un proceso puede acceder a una sección crítica específica de un programa. Esto se hace para evitar conflictos y garantizar que los recursos compartidos se utilicen de manera segura y consistente.

Semáforos

Los semáforos son una herramienta de sincronización utilizada para gestionar el acceso a recursos compartidos en sistemas operativos y programación concurrente. Funcionan como contadores que controlan el acceso a secciones críticas y permiten la exclusión mutua. Los semáforos pueden ser binarios (0 o 1) o de contadores (números enteros mayores o iguales a cero) y se utilizan para coordinar la ejecución de múltiples procesos.

De manera resumida podemos expresar:
Procesos Concurrentes: Se refiere a procesos que se ejecutan al mismo tiempo y pueden necesitar coordinarse o comunicarse entre ellos.

Condiciones de Competencia: Ocurren cuando múltiples procesos o hilos leen y escriben datos compartidos y el resultado final depende del orden en que se ejecutan las instrucciones.

Secciones Críticas: Parte del código que accede a un recurso compartido que no debe ser accedido por más de un proceso o hilo a la vez.

Exclusión Mutua: Es una propiedad que asegura que, si un proceso está ejecutando una sección crítica, entonces ningún otro proceso puede ejecutar la misma sección crítica simultáneamente.

Semáforos: Estructuras de datos que se utilizan para gestionar el acceso concurrente a recursos compartidos. Los semáforos pueden ser incrementados o decrementados para indicar si un recurso está disponible o no.

Sistemas operativos multiprogramados

Los sistemas operativos multiprogramados son aquellos que permiten que múltiples programas se ejecuten al mismo tiempo compartiendo los recursos del sistema, como la CPU, la memoria, los dispositivos de almacenamiento, entre otros. Este tipo de sistemas operativos son esenciales en la computación moderna, ya que mejoran la eficiencia y la utilización de los recursos informáticos al permitir que más de un proceso esté activo en la memoria al mismo tiempo. Aquí hay algunas características y funcionalidades clave de los sistemas operativos multiprogramados:

1. Gestión de recursos
   - CPU: Los sistemas multiprogramados utilizan técnicas de planificación de la CPU para alternar la ejecución de procesos, asegurando que la CPU siempre esté ocupada si hay tareas pendientes.
   - Memoria: Implementan gestión de memoria que permite almacenar varios programas en la memoria al mismo tiempo. Esto se realiza mediante técnicas como la paginación y la segmentación, que ayudan a aprovechar el espacio de memoria de manera eficiente.
   - Dispositivos de E/S: Los sistemas administran el acceso a dispositivos de entrada/salida para que múltiples programas puedan realizar operaciones de E/S sin interferencias indebidas entre ellos.

2. Multiplexación de tiempo
   - Los sistemas operativos multiprogramados utilizan la multiplexación de tiempo para permitir que cada proceso en ejecución tenga un poco de tiempo de CPU. Esto se hace generalmente a través de un mecanismo conocido como "time slicing" donde cada proceso recibe un pequeño intervalo de tiempo (quantum) para ejecutarse, lo que crea la ilusión de que todos los procesos se están ejecutando simultáneamente.

3. Aumento de la productividad
   - Al permitir que varios programas se ejecuten al mismo tiempo, se mejora la productividad general del sistema, ya que se puede realizar más trabajo en el mismo período de tiempo.

4. Concurrencia

- La capacidad de manejar la concurrencia es crucial en un sistema multiprogramado. Esto implica coordinar el acceso a recursos compartidos para evitar condiciones de carrera y deadlocks (interbloqueos), que pueden ocurrir cuando múltiples procesos intentan acceder al mismo recurso simultáneamente.

5. Independencia de procesos
- Los procesos en un sistema operativo multiprogramado operan de manera independiente y no deben interferir entre sí, aunque existan situaciones donde la comunicación y sincronización entre procesos sea necesaria.

Ejemplos de Sistemas Operativos Multiprogramados: Microsoft Windows, macOS, Linux, UNIX

Estos sistemas son fundamentales en entornos de servidores, computadoras personales, y dispositivos móviles, donde múltiples aplicaciones necesitan correr eficientemente al mismo tiempo.

Sistemas operativos monotarea o de tarea única

Los sistemas operativos monotarea, de tarea única o no multiprogramados son aquellos que se centran en la ejecución de un único programa o tarea a la vez. Estos sistemas son más simples en su diseño y gestión, y son comunes en entornos donde la complejidad y los recursos son limitados. Aquí algunos ejemplos de sistemas operativos no multiprogramados:

1. MS-DOS (Microsoft Disk Operating System): Uno de los sistemas operativos más conocidos que no soportaba la multiprogramación. MS-DOS ejecutaba una sola tarea a la vez, lo que significaba que los usuarios tenían que cerrar una aplicación antes de abrir otra.

2. CP/M (Control Program for Microcomputers): Antes de que MS-DOS dominara el mercado de PC, CP/M era un sistema operativo popular en computadoras personales basadas en microprocesadores. Al igual que MS-DOS, no soportaba la ejecución de múltiples programas simultáneamente.

3. Sistemas embebidos simples: Muchos sistemas operativos embebidos diseñados para tareas muy específicas en dispositivos con recursos muy limitados no admiten la multiprogramación. Estos sistemas están optimizados para ejecutar un conjunto limitado de tareas con requisitos mínimos de procesamiento y memoria. Ejemplos incluyen firmware en dispositivos como microcontroladores en electrodomésticos, sistemas de control industrial, y otros dispositivos de hardware dedicado.

4. Sistemas de tiempo real: Algunos sistemas operativos de tiempo real (RTOS) diseñados para aplicaciones muy específicas pueden no implementar multiprogramación si la predictibilidad y la respuesta inmediata son más críticas que la multitarea. Por ejemplo, sistemas en vehículos aeroespaciales o médicos que gestionan una tarea crítica a la vez.

5. Apple DOS: Utilizado en las primeras computadoras Apple, este sistema operativo también estaba diseñado para ejecutar una sola tarea o programa a la vez.

Estos sistemas operativos tienen en común la simplicidad de diseño y la limitación en la ejecución concurrente de múltiples aplicaciones, lo que los hace adecuados para dispositivos con restricciones severas de hardware o para aplicaciones que requieren una gestión muy controlada y predecible del procesamiento.