Fundamentos de Sistemas Operativos y Administración de Procesos

De la teoría al funcionamiento real de los SO modernos

Objetivos del curso

1 **Explicar el origen y la evolución de los sistemas operativos**

Comprender cómo los sistemas operativos se han transformado de la operación manual a los sistemas distribuidos complejos

2 **Analizar las funciones, los componentes y los tipos de SO**

Examinar los componentes críticos que componen los sistemas operativos modernos y sus diversas clasificaciones

3 **Comprender la inicialización y la administración del SO**

Aprender el proceso de arranque y cómo los sistemas operativos administran los recursos de hardware

4 **Aplicar los conceptos de administración de procesos y comparar el SO actual**

Dominar los principios de administración de procesos y evaluar las diferencias entre los sistemas operativos contemporáneos

Evolución de los sistemas operativos

1

Primera generación (décadas de 1940 y 1950)

Sin sistemas operativos. Las computadoras funcionaban manualmente a través de interruptores y tarjetas perforadas. Los programas se ejecutan uno a la vez con manipulación directa del hardware.

2

Segunda generación (décadas de 1950 y 1960)

Introducción de sistemas de procesamiento por lotes. Los monitores residentes simples permitieron la secuenciación automática de trabajos, lo que redujo el tiempo de inactividad entre programas.

3

Tercera generación (décadas de 1960 a 1980)

Aparición de la multitarea y la multiprogramación. Varios programas podían residir en la memoria simultáneamente, optimizando el uso de la CPU.

4

Cuarta generación (décadas de 1980 y 1990)

Desarrollo de interfaces gráficas de usuario y capacidades de red. Las computadoras personales se generalizaron con Windows y Macintosh OS.

5

Era actual (2000-presente)

Sistemas distribuidos, sistemas operativos móviles, virtualización y computación en la nube. Centrarse en la seguridad, la escalabilidad y la optimización de recursos.

Funciones principales de un sistema operativo

Gestión de procesos

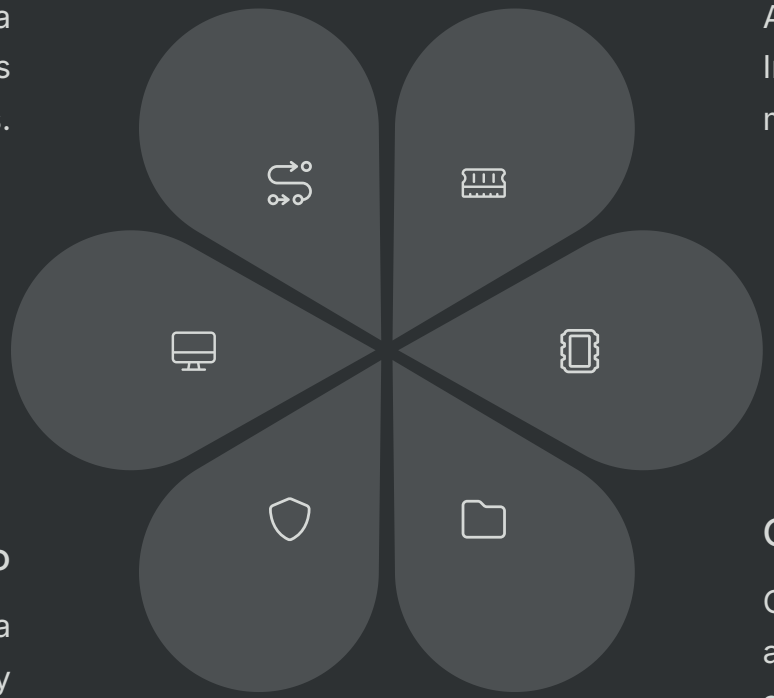
Crea, programa y finaliza procesos. Asigna tiempo de CPU y gestiona los estados y las transiciones de los procesos.

Interfaz de usuario

Proporciona mecanismos de interacción para usuarios y aplicaciones. Incluye interfaces gráficas y de línea de comandos.

Seguridad y control de acceso

Protege los recursos del sistema. Implementa mecanismos de autenticación, autorización y cifrado.



Administración de memoria

Asigna y desasigna espacio de memoria. Implementa memoria virtual, paginación y mecanismos de segmentación.

Gestión de dispositivos de E/S

Controla los dispositivos periféricos a través de controladores. Gestiona la comunicación del dispositivo y los búferes de transferencia de datos.

Gestión del sistema de archivos

Organiza y mantiene las estructuras de archivos. Controla la creación, eliminación, acceso y asignación de almacenamiento de archivos.

Tipos de sistemas operativos

Por manejo de tareas

- Monotarea: ejecuta solo un programa a la vez (DOS)
- Multitarea: ejecuta varios programas simultáneamente (SO moderno)

Por número de usuarios

- Un solo usuario: diseñado para uso individual (Windows antiguo)
- Multiusuario: admite varios usuarios simultáneos (UNIX, Linux)

Por tiempo de respuesta

- Tiempo compartido: asigna porciones de tiempo de CPU a los procesos
- Tiempo real: garantiza la respuesta dentro de las restricciones de tiempo especificadas

Por arquitectura

- Distribuido: funciona en varias máquinas físicas
- Integrado: integrado en dispositivos con funciones específicas
- Móvil: optimizado para dispositivos portátiles con interfaces táctiles

Proceso de arranque del sistema operativo

Autocomprobación al encender (POST)

BIOS/UEFI realiza diagnósticos de hardware para garantizar que todos los componentes funcionen correctamente. Verifica la RAM, la CPU y los periféricos esenciales.

Ejecución del gestor de arranque

GRUB (Linux) o el Administrador de arranque de Windows se cargan desde MBR/GPT. Presenta opciones del sistema operativo y carga archivos de configuración.

Carga del kernel

El kernel del sistema operativo se carga en la memoria. Comienza la detección de hardware y se inicializan los controladores principales del sistema.

Inicialización del servicio

Los servicios del sistema y los demonios se inician en secuencia. En Linux, systemd o el proceso init gestiona esta fase; en Windows, el Administrador de control de servicios lo gestiona.

Entorno de usuario

Aparece el administrador de inicio de sesión, autentica al usuario e inicia el entorno de escritorio o la interfaz de comandos.

Arquitectura interna de los sistemas operativos

Tipos de kernel

Kernel monolítico

Todos los servicios del sistema operativo se ejecutan en el espacio del kernel. Ejemplos: Linux, UNIX tradicional. Ejecución más rápida pero menos aislamiento de fallos.

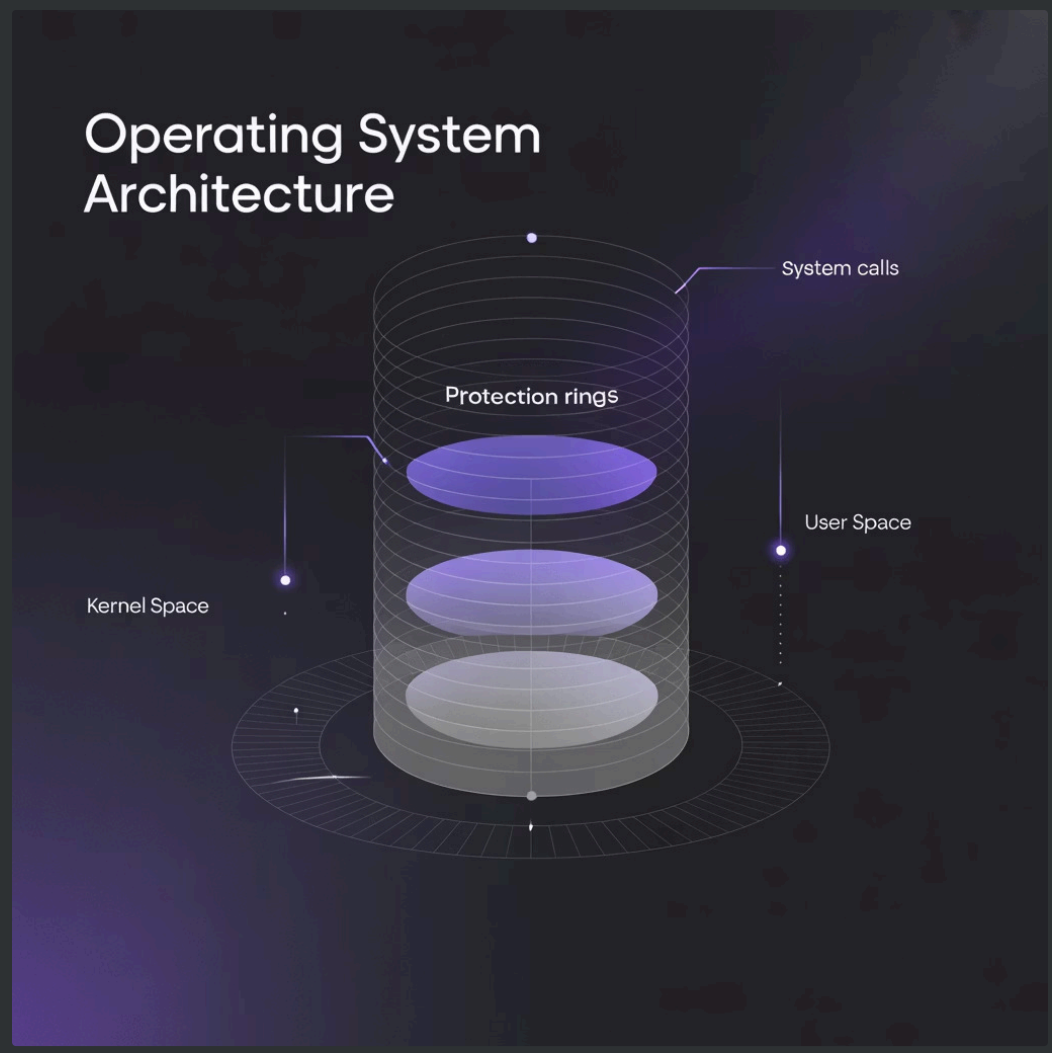
Microkernel

Kernel mínimo con servicios en el espacio de usuario. Ejemplos: QNX, MINIX. Mejor estabilidad pero potencialmente más lento.

Kernel híbrido

Combina aspectos de ambos enfoques. Ejemplos: Windows NT, macOS. Equilibrio entre rendimiento y modularidad.

Modos de operación



Modo Kernel

Modo privilegiado con acceso ilimitado al hardware. Ejecuta funciones críticas del sistema y tiene acceso completo a la memoria.

Modo de usuario

Entorno restringido para aplicaciones. Acceso limitado a los recursos del sistema a través de llamadas API controladas.

Administración de la memoria

Técnicas de organización de la memoria

1

Segmentación

Divide la memoria en segmentos de diferentes tamaños según las divisiones lógicas de los programas. Cada segmento tiene una dirección base y un límite.

2

Paginación

Divide la memoria física y virtual en bloques de tamaño fijo llamados páginas. Asignación gestionada a través de tablas de páginas.

3

Memoria virtual

Crea la ilusión de más memoria de la que está disponible físicamente. Permite que los programas utilicen direcciones independientes de la ubicación física de la memoria.

Desafíos de la administración de la memoria

1

Fragmentación

- Interna: espacio desperdiciado dentro de los bloques asignados
- Externa: espacios inutilizables entre los bloques asignados

2

Intercambio

Proceso de mover páginas de memoria entre la RAM y el almacenamiento en disco cuando la memoria física está llena.

Gestión de entrada/salida

1

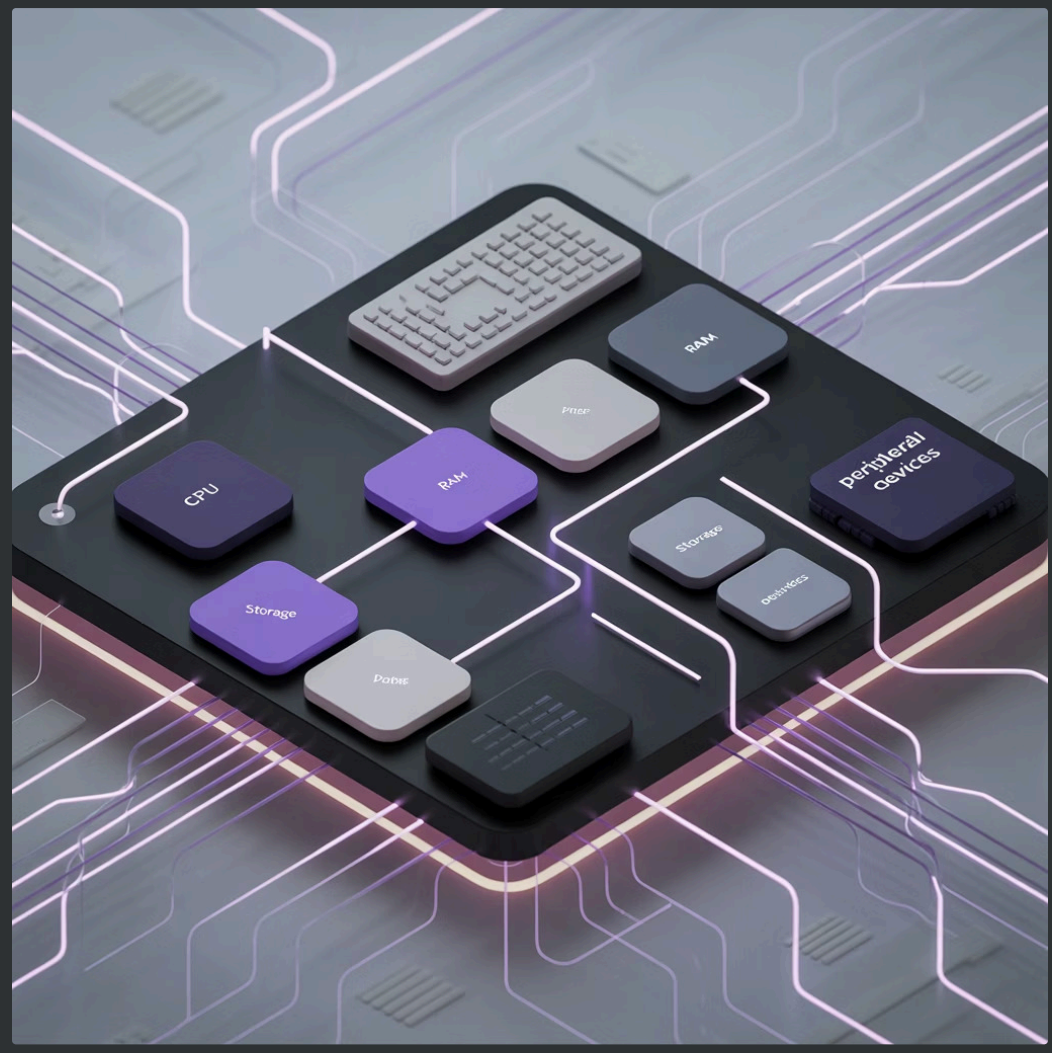
Controladores de dispositivos

Módulos de software que sirven de interfaz entre los dispositivos de hardware y el sistema operativo. Traducen los comandos genéricos de E/S en operaciones específicas del dispositivo.

2

Métodos de acceso a E/S

- Sondeo: El sistema operativo comprueba continuamente el estado del dispositivo
- Interrupciones: El dispositivo señala a la CPU cuando se completa la operación
- DMA (Acceso directo a la memoria): Transferencia de datos sin intervención de la CPU



1

Técnicas de transferencia de datos

- Almacenamiento en búfer: Área de almacenamiento temporal para los datos en tránsito
- Spooling: Mecanismo de colas para la salida de la impresora y otros dispositivos
- Caché: Memoria de alta velocidad para los datos a los que se accede con frecuencia

Comprensión de los procesos

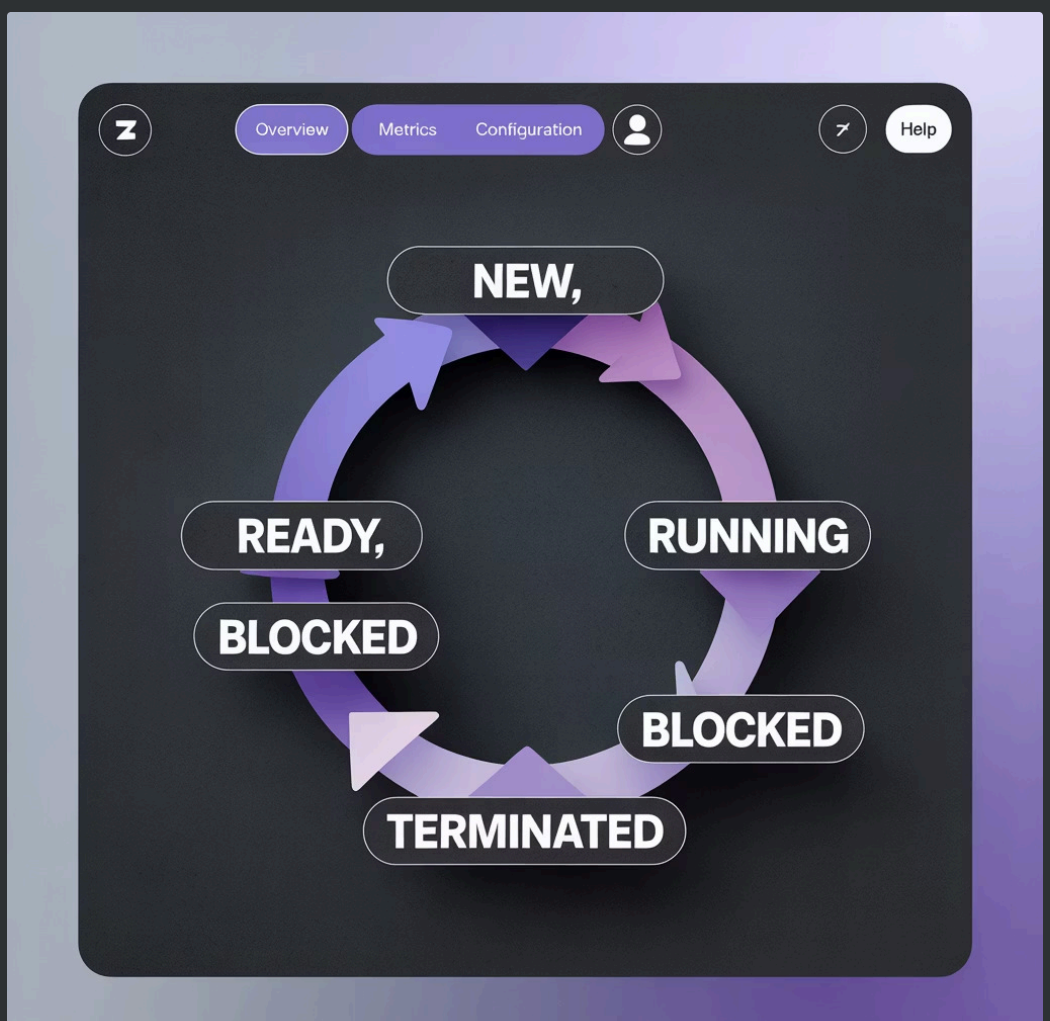
Definición de proceso

Un proceso es un programa en ejecución. Incluye el código del programa (sección de texto), la actividad actual (contador de programa, registros), la pila (datos temporales), el montón (memoria asignada dinámicamente) y la sección de datos (variables globales).

Bloque de control de procesos (PCB)

- ID y estado del proceso
- Contador de programa y registros de CPU
- Información de programación de la CPU
- Información de gestión de memoria
- Información del estado de E/S
- Información contable

Estados del proceso



Nuevo

El proceso se está creando

Listo

Esperando ser asignado a un procesador

En ejecución

Las instrucciones se están ejecutando

Bloqueado/En espera

Esperando que ocurra un evento

Terminado

La ejecución ha finalizado

Proceso vs Hilo

Características del proceso

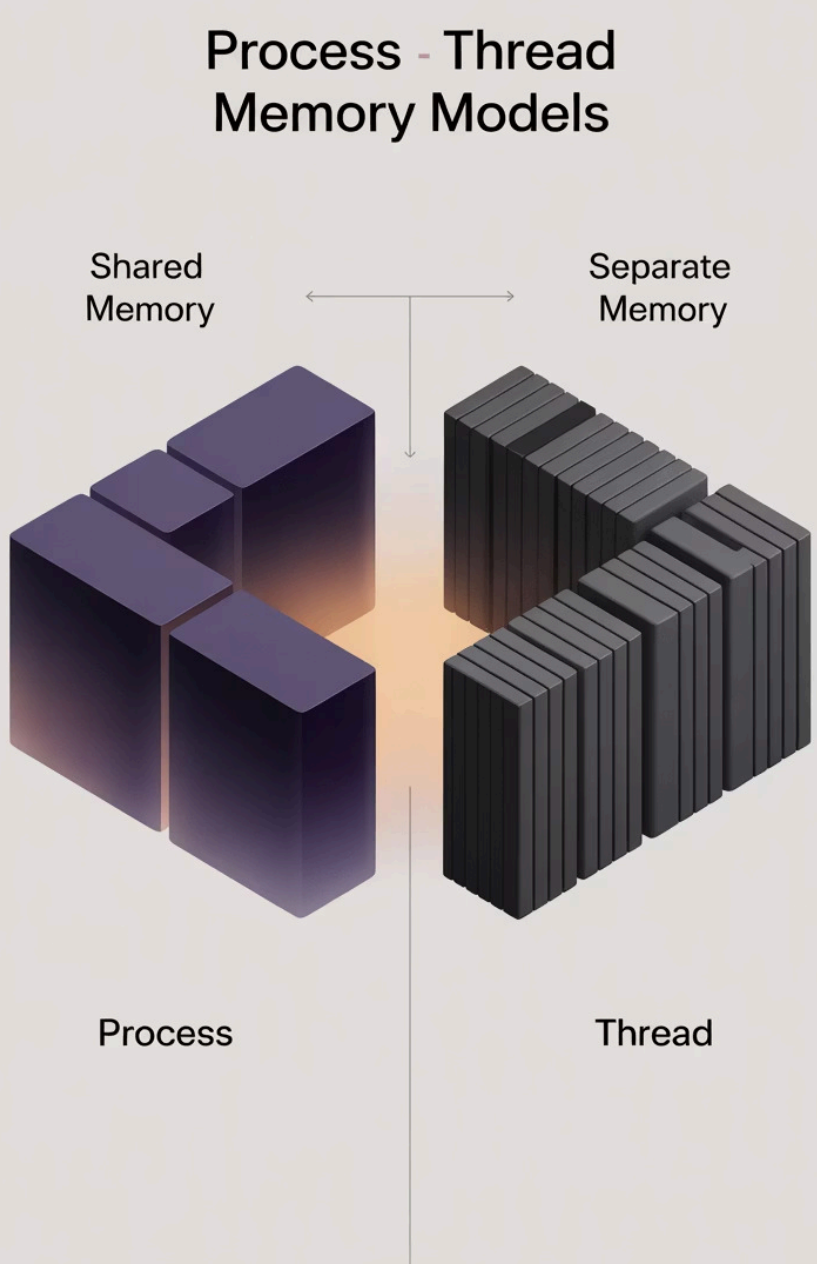
- Entidad de ejecución independiente
- Tiene su propio espacio de memoria
- Requisitos de recursos más pesados
- La comunicación entre procesos es más compleja
- El cambio de contexto es más caro

Ejemplos: Diferentes aplicaciones que se ejecutan simultáneamente como un navegador web y un procesador de textos

Características del hilo

- Unidad de ejecución ligera dentro de un proceso
- Comparte espacio de memoria con otros hilos en el mismo proceso
- Requiere menos recursos para crear y mantener
- La comunicación a través de la memoria compartida es más sencilla
- El cambio de contexto es más rápido

Ejemplos: Varias pestañas en un navegador web, revisión ortográfica en segundo plano en un procesador de textos



Programación de Procesos

1

Tipos de Programador

- A largo plazo: Controla el grado de multiprogramación seleccionando qué procesos entran en el sistema
- A medio plazo: Gestiona el intercambio de procesos entre la memoria principal y el disco
- A corto plazo: Selecciona qué proceso listo obtiene el siguiente tiempo de CPU

Algoritmos de Programación Comunes

Primero en Llegar, Primero en Ser Servido (FCFS)

Los procesos se ejecutan en orden de llegada. Simple pero puede llevar al "efecto convoy" donde los procesos cortos esperan detrás de los largos.

Trabajo Más Corto Primero (SJF)

Selecciona el proceso con el tiempo de ejecución esperado más corto. Óptimo para el tiempo de espera promedio, pero requiere conocer el tiempo de ejecución de antemano.

Round Robin (RR)

Asigna a cada proceso una porción de tiempo (quantum) en un orden circular. Distribución justa, pero el rendimiento depende de la selección del tamaño del quantum.

Programación por Prioridades

Asigna un valor de prioridad a cada proceso y selecciona primero la prioridad más alta. Puede causar inanición para los procesos de menor prioridad.

Comparación de algoritmos de programación

Primero en llegar, primero en ser servido (FCFS)

Secuencia de ejecución del proceso: P1, P2, P3, P4

Proceso	Tiempo de espera
P1 (10ms)	0ms
P2 (5ms)	10ms
P3 (8ms)	15ms
P4 (3ms)	23ms
Promedio	12ms

El trabajo más corto primero (SJF)

Secuencia de ejecución del proceso: P4, P2, P3, P1

Proceso	Tiempo de espera
P4 (3ms)	0ms
P2 (5ms)	3ms
P3 (8ms)	8ms
P1 (10ms)	16ms
Promedio	6.75ms

Round Robin (quantum = 4ms) ejecutaría fragmentos de cada proceso por turnos, equilibrando la capacidad de respuesta pero aumentando los cambios de contexto.

Comunicación entre procesos (IPC)

Métodos de comunicación

Tuberías

Canal de datos unidireccional entre procesos relacionados. Ejemplo: canalización de línea de comandos en Unix (ls | grep)

Colas de mensajes

Listas enlazadas de mensajes almacenados dentro del kernel. Los procesos pueden leer/escribir mensajes independientemente de otros procesos.

Memoria compartida

Múltiples procesos acceden a la misma región de memoria. El método IPC más rápido, pero requiere sincronización.

Sockets

Puntos finales de comunicación que pueden conectar procesos a través de límites de red.

Mecanismos de sincronización

Semáforos

Variables enteras para la señalización entre procesos. Se utiliza para controlar el acceso a los recursos compartidos.

Mutexes

Semáforos binarios que protegen las secciones críticas. Solo un proceso puede mantener el bloqueo a la vez.

Monitores

Construcciones de sincronización de alto nivel que encapsulan tanto datos como procedimientos.

Condiciones de carrera y secciones críticas

El problema

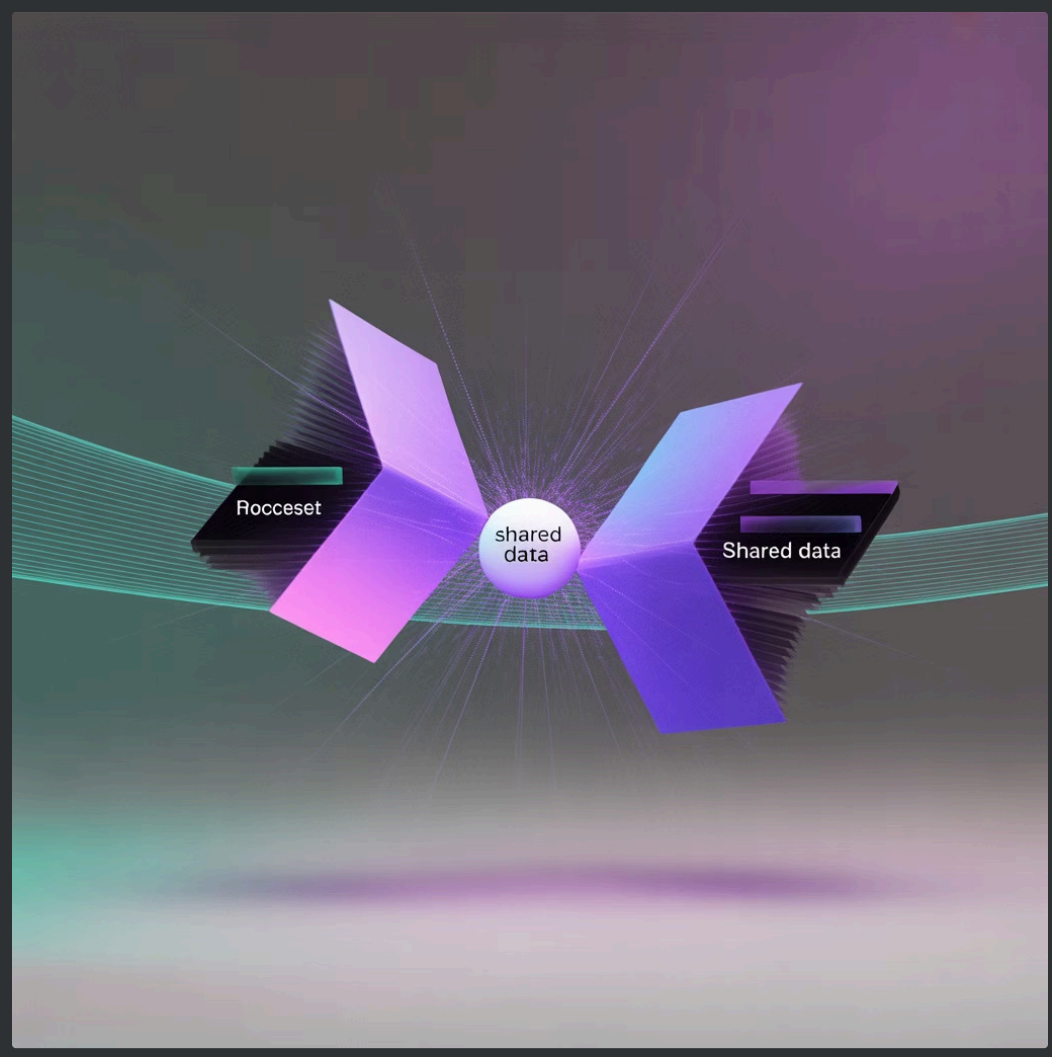
Las condiciones de carrera ocurren cuando varios procesos acceden y manipulan datos compartidos simultáneamente, y el resultado depende del orden particular de ejecución.

Ejemplo del mundo real

Dos procesos de cuentas bancarias que intentan retirar dinero simultáneamente. Sin sincronización, ambos podrían leer el saldo inicial antes de que ninguno lo actualice, lo que podría permitir retiros que excedan el saldo disponible.

Requisitos de la sección crítica

- Exclusión mutua: Solo un proceso en la sección crítica a la vez
- Progreso: Los procesos fuera de la sección crítica no pueden bloquear a otros
- Espera limitada: El proceso no puede esperar indefinidamente




```
// Pseudocódigo para el problema de la sección crítica
Proceso P1:
 while (true) {
   entry_section();    // Solicitar acceso

   // Sección crítica
   shared_variable++;  // Vulnerable a la carrera

   exit_section();     // Liberar acceso

   // Sección restante
 }
```

Sin la sincronización adecuada, las operaciones como shared_variable++ no son atómicas y pueden conducir a resultados inconsistentes cuando se ejecutan simultáneamente.

Comparación entre GNU/Linux y MS Windows

Característica	GNU/Linux	MS Windows
Filosofía	Código abierto, diseño modular, desarrollo impulsado por la comunidad	Propietario, modelo de desarrollo centralizado
Modelo de seguridad	Permisos controlados por el usuario, parches de seguridad transparentes	Seguridad automatizada con menos control del usuario, actualizaciones regulares
Sistemas de archivos	ext4, XFS, Btrfs, ZFS (estructura jerárquica)	NTFS, ReFS (utiliza letras de unidad)
Interfaz de usuario	Múltiples entornos de escritorio (GNOME, KDE, etc.), potente CLI	GUI estandarizada, CLI limitada (PowerShell mejorando)
Gestión de procesos	Procesos altamente visibles, control de prioridad flexible	Gestión de procesos más abstraída, simplificada para los usuarios
Usos comunes	Servidores, superordenadores, educación, sistemas integrados	Informática de escritorio, aplicaciones empresariales, juegos
Tipo de kernel	Monolítico (con módulos cargables)	Híbrido (kernel NT)

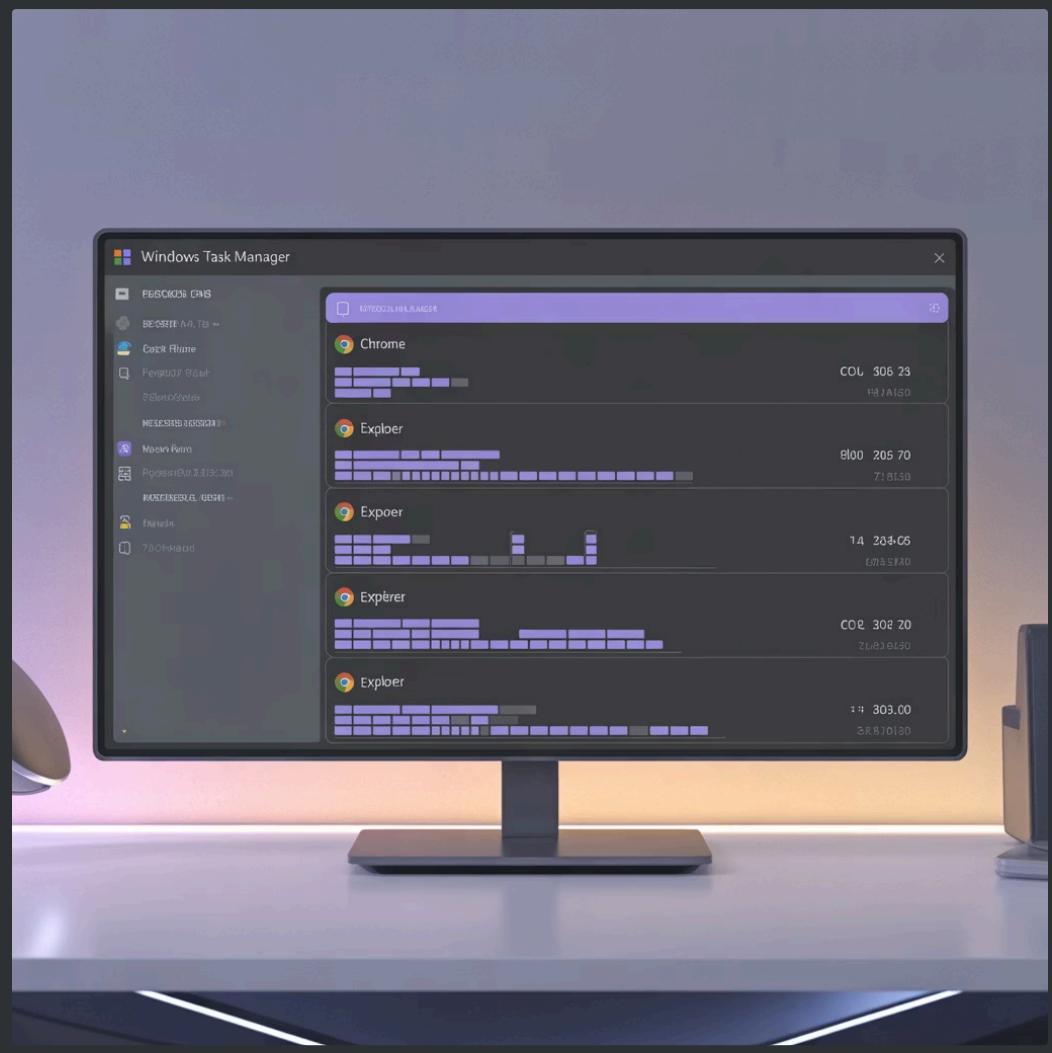
Aplicaciones prácticas: Monitoreo de procesos

Gestión de procesos de Linux

```
# Ver todos los procesos
$ ps aux

# Visor de procesos interactivo
$ top

# Alternativa moderna a top
$ htop

# Matar un proceso
$ kill -9 PID

# Cambiar la prioridad del proceso
$ renice -n 10 -p PID
```

Linux proporciona amplias herramientas de línea de comandos para la inspección y gestión de procesos, lo que brinda a los administradores un control granular sobre los recursos del sistema.

Gestión de procesos de Windows



El Administrador de tareas de Windows proporciona una interfaz gráfica para monitorear los procesos, con opciones para:

- Ver el uso de CPU, memoria, disco y red
- Finalizar tareas y procesos
- Establecer la prioridad del proceso (clic derecho → Establecer prioridad)
- Ver los detalles y las dependencias del proceso

Los comandos de PowerShell como Get-Process y Stop-Process ofrecen alternativas programables.

Herramientas de monitorización de memoria y CPU

Herramientas de monitorización de Linux

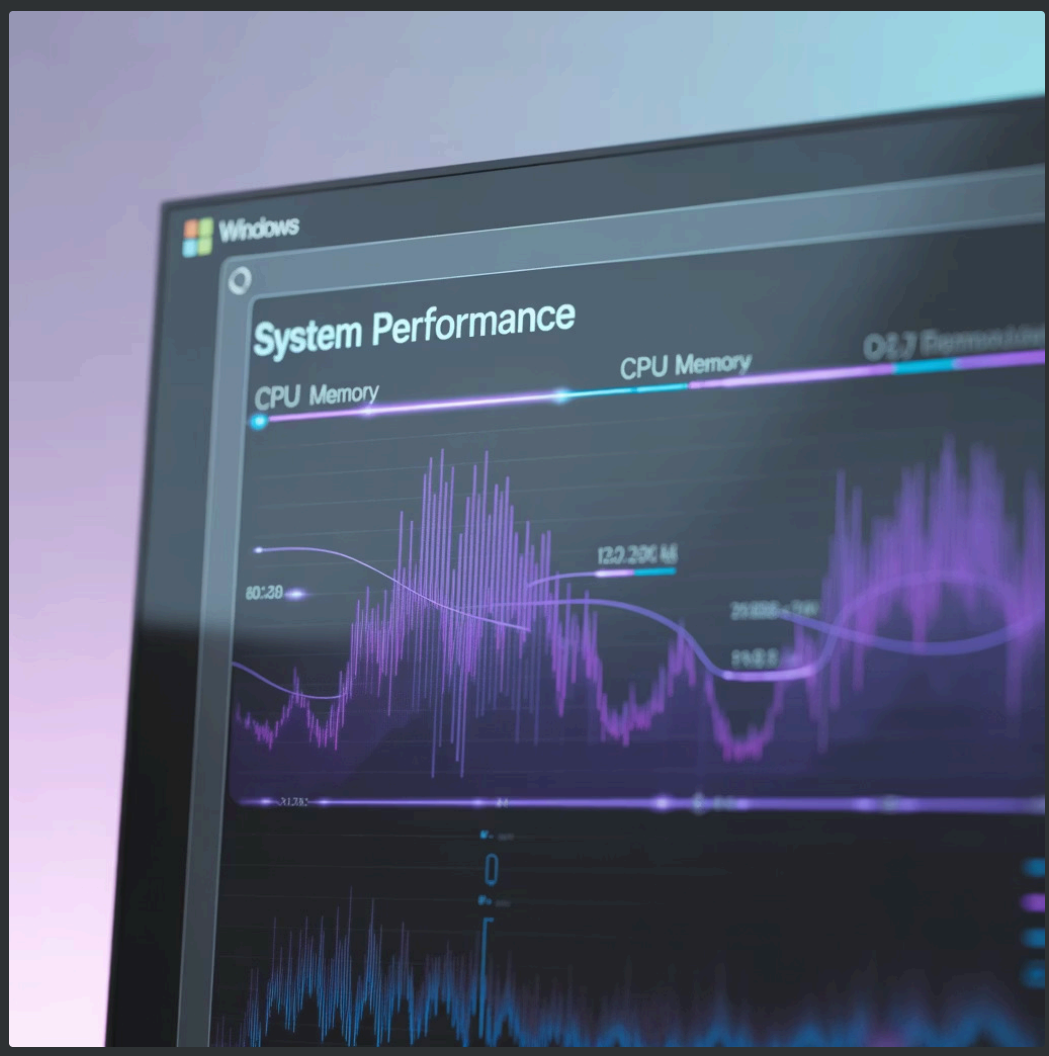
Monitorización de memoria

- free: Muestra la cantidad de memoria libre y utilizada
- vmstat: Estadísticas de memoria virtual
- pmap: Mapa de memoria del proceso

Monitorización de CPU

- mpstat: Estadísticas de múltiples procesadores
- iostat: Estadísticas de CPU y E/S
- sar: Informador de actividad del sistema

Herramientas de monitorización de Windows



Monitor de rendimiento

Herramienta avanzada para el seguimiento detallado de recursos con contadores personalizables y capacidades de registro.

Monitor de recursos

Vista más detallada que el Administrador de tareas con gráficos en tiempo real y opciones de filtrado para procesos.

Conceptos avanzados de procesos

Programación de procesos en sistemas multinúcleo

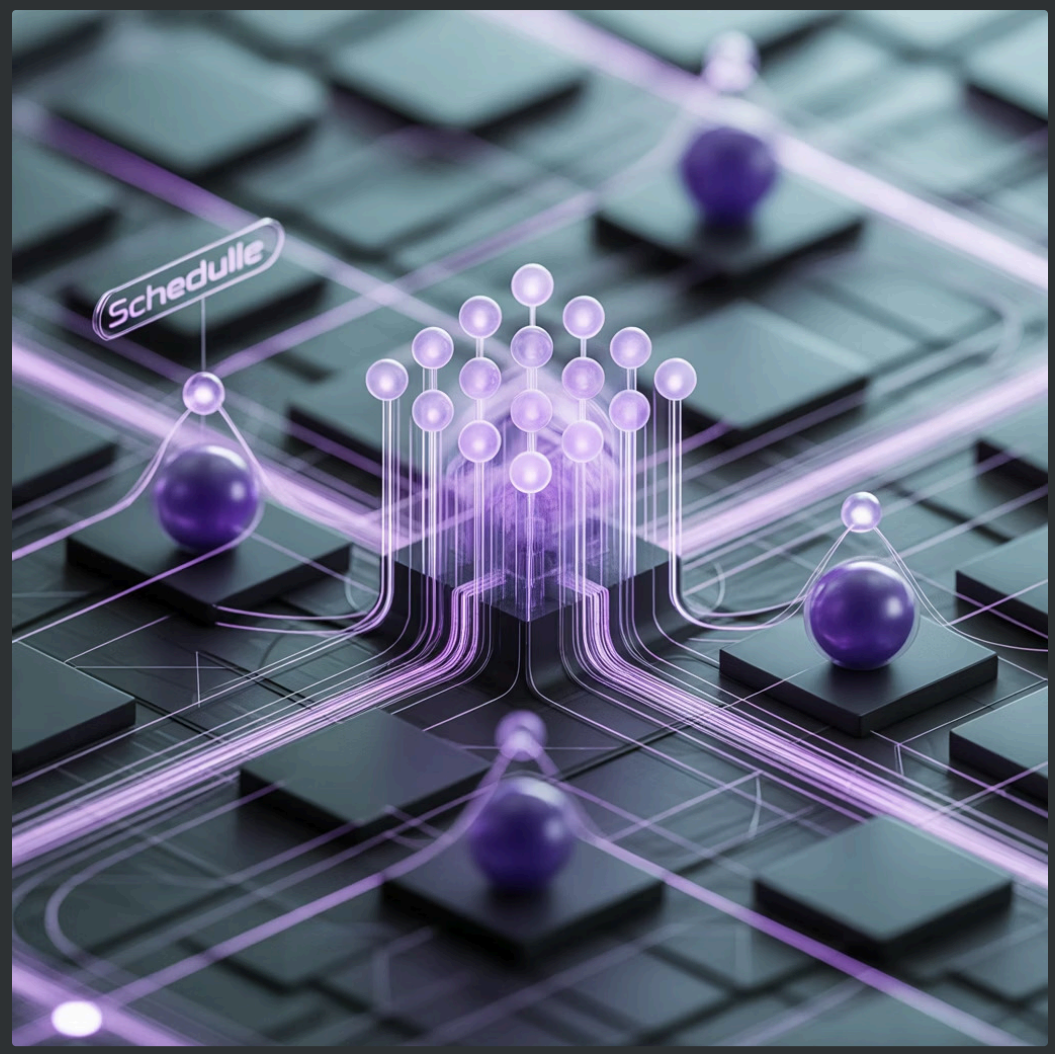
Los sistemas operativos modernos implementan algoritmos de programación sofisticados para utilizar eficientemente múltiples núcleos de CPU:

- Equilibrio de carga entre núcleos
- Afinidad de CPU para optimizar el uso de la caché
- Conciencia de NUMA (acceso no uniforme a la memoria)
- Programación con reconocimiento de energía para gestionar el consumo de energía

Programación en tiempo real

Algunos sistemas operativos admiten la programación en tiempo real para aplicaciones de tiempo crítico:

- Tiempo real estricto: plazos absolutos (sistemas integrados)
- Tiempo real flexible: plazos preferidos pero flexibles (multimedia)



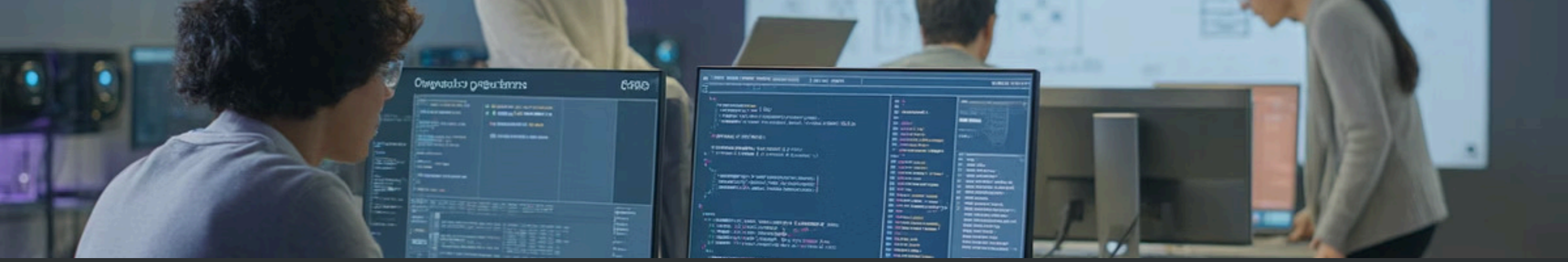
Virtualización y contenedorización

La gestión moderna de procesos se extiende a entornos virtualizados:

- Máquinas virtuales con sus propias instancias de SO aisladas
- Contenedores que comparten el kernel del host pero con procesos aislados
- Hipervisores que gestionan la asignación de recursos entre instancias

Tendencia de la industria

Las plataformas de orquestación de contenedores como Kubernetes han revolucionado la gestión de procesos en entornos de nube al automatizar la implementación, el escalado y las operaciones de los contenedores de aplicaciones.



Conclusiones clave

Evolución del SO

Los sistemas operativos han evolucionado desde simples procesadores por lotes hasta plataformas complejas y distribuidas que gestionan diversos recursos de hardware y proporcionan experiencias de usuario enriquecidas.

Funcionalidad principal

Todos los sistemas operativos proporcionan servicios fundamentales: gestión de procesos, asignación de memoria, control de E/S, sistemas de archivos y mecanismos de seguridad.

Gestión de procesos

La programación, sincronización y comunicación eficaces de los procesos son cruciales para el rendimiento del sistema, la utilización de los recursos y la prevención de condiciones de carrera