Sistemas Operativos

Kernel

- Versión: Marzo 2025
- Palabras Claves: Sistemas Operativos, Harware, Kernel, llamadas al sistema

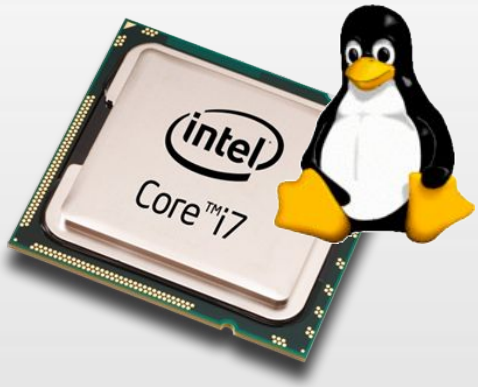
Los temas vistos en estas diapositivas han sido mayormente extraídos del libro de William Stallings (Sistemas Operativos: Aspectos internos y principios de diseño) y Conceptos de sistemas operativos (Silberschatz, Galvin, Gagne)

¿Qué es un Sistema Operativo?

¿SO?

Sistema Operativo

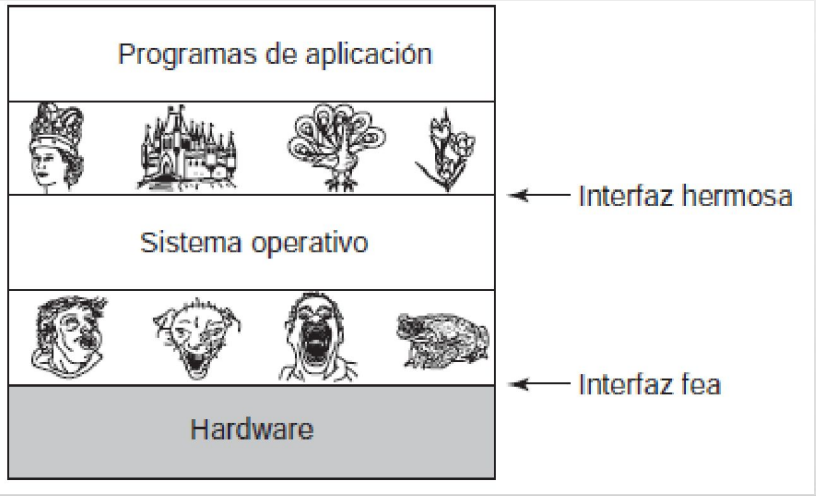
- Un sistema operativo es un software que actúa como intermediario entre el usuario de una computadora y su hardware.



- Es **software**:
  - ✔ Necesita procesador y memoria para ejecutarse

Sistema Operativo

- Gestiona el HW
- Controla la ejecución de los **procesos**
- Interfaz entre aplicaciones y HW
- Actúa como intermediario entre un usuario de una computadora y el HW de la misma



Dos perspectivas o miradas

- ✔ Desde el usuario (de arriba hacia abajo)
- ✔ Desde el sistema (de abajo hacia arriba)

Perspectiva desde el usuario

- Abstracción con respecto a la arquitectura
  - Arquitectura: conjunto de instrucciones, organización de memoria, E/S, estructura de bus)
- El SO "oculta" el HW y presenta a los programas abstracciones más simples de manejar.
- Los programas de aplicación son los "clientes" del SO.
- Comparación: uso de escritorio y uso de comandos de texto
- Comodidad, "amigabilidad" (friendliness)

Perspectiva desde la administración de recursos

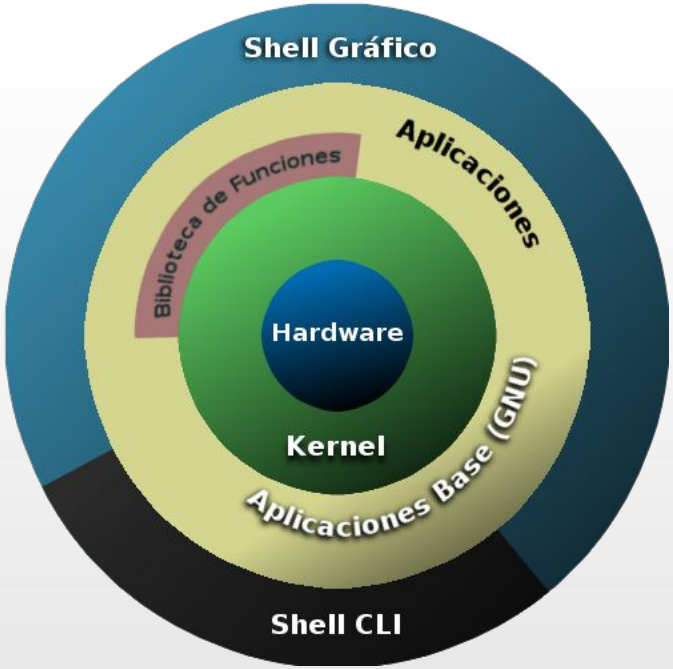
- Administra los recursos de HW de uno o más **procesos**
- Provee un conjunto de servicios a los usuarios del sistema
- Maneja la memoria secundaria y dispositivos de I/O (Input/Output – Entrada/Salida)
- Ejecución simultánea de procesos
- Multiplexación en tiempo (CPU) y en espacio (memoria)

Objetivos de los S.O.

- Comodidad
  - ✔ Hacer mas fácil el uso del hardware (PC, servidor, switch, router, controlador específico)
- Eficiencia
  - ✔ Hacer un uso más eficiente de los recursos del sistema
- Evolución
  - ✔ Permitir la introducción de nuevas funciones al sistema sin interferir con funciones anteriores

Componentes de un SO

- Kernel
- Shell
  - GUI / CLI



- Herramientas
  - Editores, Compiladores, Librerías, etc.

Kernel (Núcleo)

- "Porción de código"
  - ✔ Se encuentra en memoria principal
  - ✔ Se encarga de la administración de los recursos.
- Implementa servicios esenciales:
  - ✔ Manejo de memoria
  - ✔ Manejo de la CPU
  - ✔ Administración de procesos
  - ✔ Comunicación y Concurrencia
  - ✔ Gestión de la E/S

Kernel - ¿Qué es el kernel Linux?

- Programa que ejecuta programas y gestiona dispositivos de hardware
- Encargado de que el software y el hardware puedan trabajar juntos
- Principales funciones
  - Administración de memoria principal
  - Administración de uso de la CPU
- Es de código abierto a los usuarios (kernel/sched.c)
- En una misma estructura de código fuente se da soporte a todas las arquitecturas
- Liberado bajo licencia GPLv2
- En un sentido estricto es el Sistema Operativo

Kernel - ¿Qué es el kernel Linux?

- Es responsable de facilitar a los procesos acceso seguro al hardware
- Para ello, utiliza una interfaz conocida como "llamadas al sistema" (lo vemos más adelante)
- Los procesos de usuario son clientes del Sistema Operativo
- El Kernel gestiona y atiende los requerimientos de los distintos procesos siguiendo un criterio de "equidad"

Kernel - ¿Qué es el kernel Linux?

- El Kernel se ejecuta en modo supervisor o privilegiado
  - En este modo se tiene acceso al conjunto completo de instrucciones que permiten, entre otras cosas:
    - Acceder al hardware
    - Direccionar la memoria
    - Programar la CPU
    - Otras…
- Los procesos se ejecutan en modo usuario
  - Cuando un proceso requiere acceso al hardware, lo hace a través de una llamada al sistema operativo “System Call”
- La capacidad de ejecutar código en modo supervisor o modo usuario, es provista por el hardware que trabaja en conjunto son el Sistema Operativo

Apoyo del Hardware

- Modos de Ejecución: Define limitaciones en el conjunto de instrucciones que se puede ejecutar en cada modo
- Interrupción de Clock: Se debe evitar que un proceso se apropie de la CPU
- Protección de la Memoria: Se deben definir límites de memoria a los que puede acceder cada proceso (registros base y límite)

Modos de ejecución

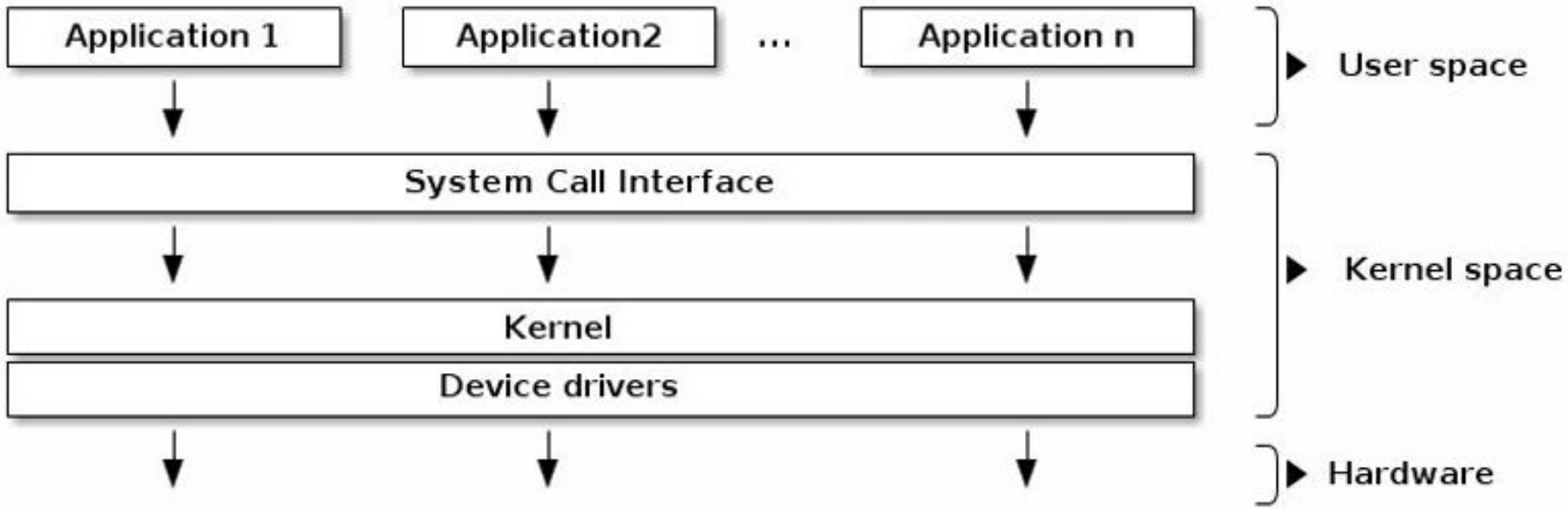
- El bit en la CPU indica el modo actual
- Las instrucciones privilegiadas deben ejecutarse en modo **Supervisor o Kernel**
  - Necesitan acceder a estructuras del kernel, o ejecutar código que no es del proceso
- En modo **Usuario**, el proceso puede acceder sólo a su espacio de direcciones, es decir a las direcciones "propias".

Bit de modo

- Cuando se arranque el sistema, arranca con el bit en modo supervisor.
- Cada vez que comienza a ejecutarse un proceso de usuario, este bit se DEBE PONER en modo usuario.
  - Mediante una Interrupción
- Cuando hay un trap o una interrupción, el bit de modo se pone en modo Kernel.
  - Única forma de pasar a Modo Kernel
  - No es el proceso de usuario quien hace el cambio explícitamente.

Kernel - Funcionalidad del kernel

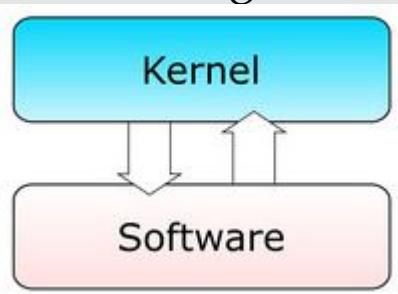
Arquitectura típica:



Fuente: https://linux-kernel-labs.github.io

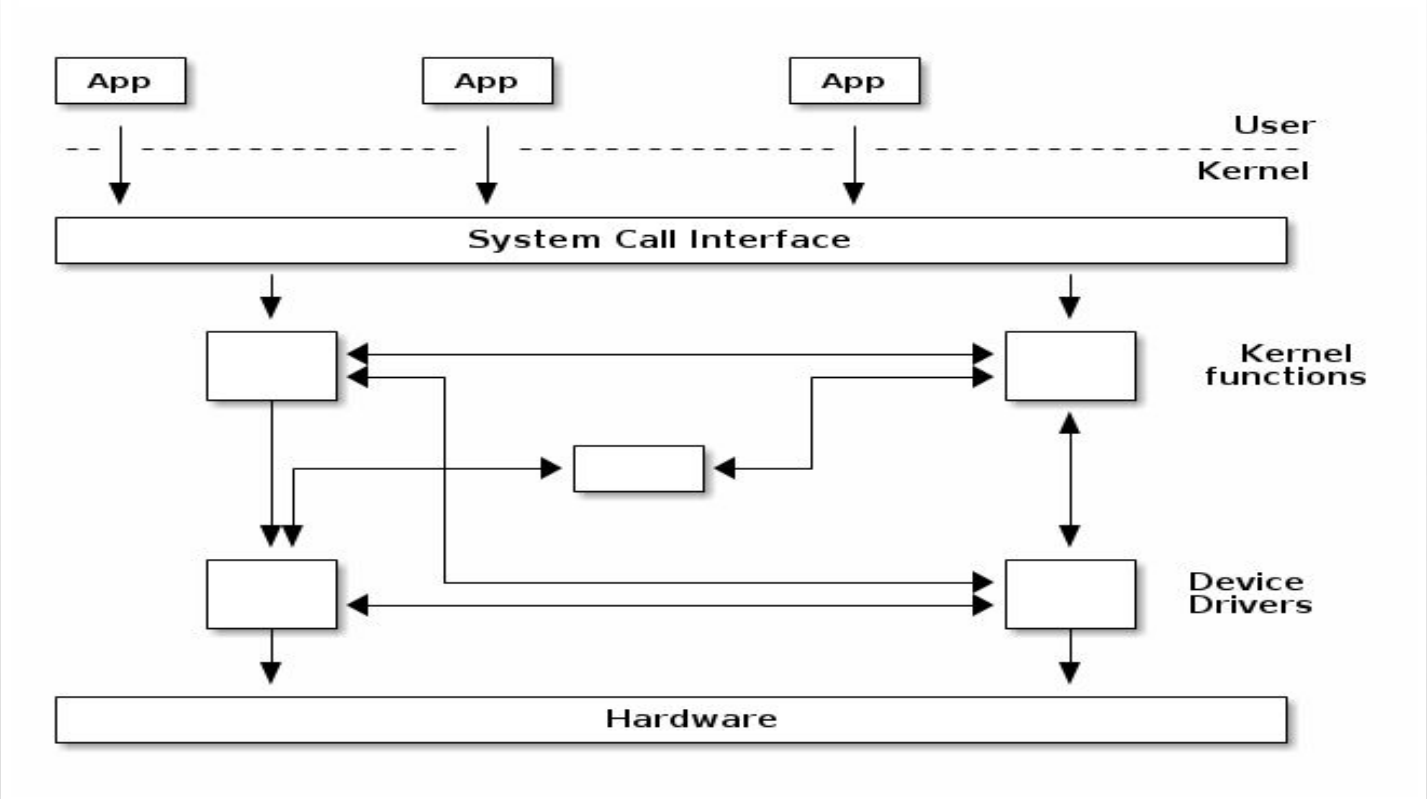
Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Monolítico:**
    - Incluye todos los servicios del sistema operativo en un solo bloque de código que se ejecuta en modo supervisor (gestión de procesos, memoria, archivos, controladores, etc.)
    - Posee distintos subsistemas y ka funcionalidad de cada uno es accedida directamente desde otro a través de funciones públicas
    - Gestión completa de recursos (memoria, cpu, E/S)
    - Posee acceso completo a los recursos de hardware Eficiencia al no requerir cambios de modo mientras se ejecuta
    - Manejo directo de interrupciones y excepciones
    - Permite opcionalmente un enfoque modular, pero los módulos siguen ejecutándose en modo Kernel GNU/Linux
    - Al ser un único bloque de código, es más complejo de manejar, administrar y mantener
    - Linux, Unix, FreeBSD



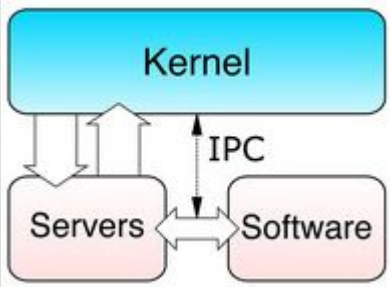
Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Monolítico:**



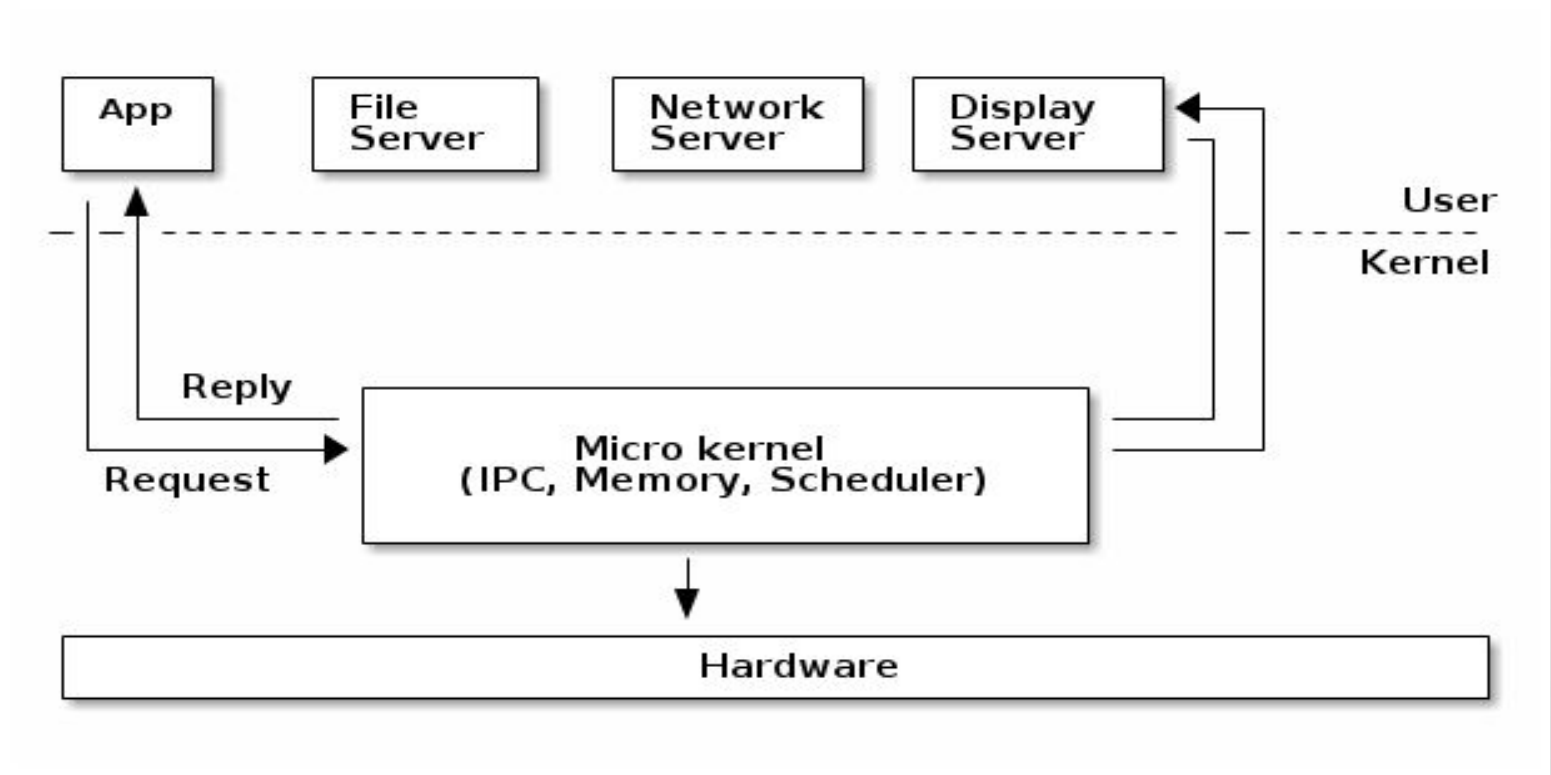
Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Microkernel:**
    - Minimiza la cantidad de código que se ejecuta en modo supervisor con el fin de hacerlo más liviano respecto a un monolítico
    - Incluye funciones más esenciales, como la gestión de procesos, comunicación entre procesos y gestión de memoria. Otros servicios (como controladores de hardware, sistemas de archivos y red) se ejecutan en espacio de usuario
    - El altamente modular permitiendo su personalización a través de la adición o eliminación de módulos de funcionalidad
    - Dado que la mayoría de los servicios se ejecutan en modo usuario, suele ser más estable y seguro que los monolíticos
    - Provee un rendimiento inferior por los cambios de modo constantes que requiere para su ejecución
    - Su desarrollo y mantenimiento es más eficiente dado a la separación de funciones
    - Minix y QNX, L4



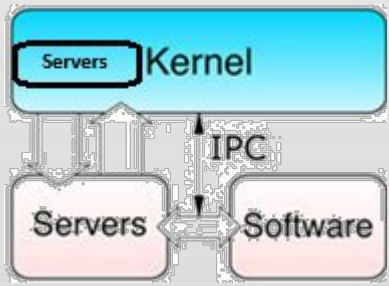
Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Microkernel:**



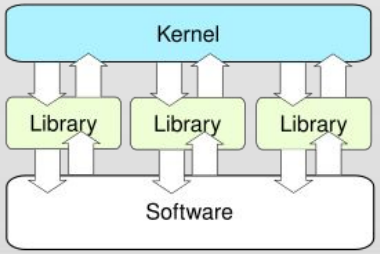
Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Híbrido:**
    - Combina características de los Kernels monolíticos y Microkernels
    - Tiene un núcleo más pequeño que un monolítico. Incluye algunos servicios en modo núcleo (eficiencia) y otros se ejecutan en modo usuario (seguridad).
    - Son modulares. Ofrecen un equilibrio entre rendimiento y modularidad lo que permite actualizaciones más sencillas y una mayor flexibilidad y facilidad a su desarrollo.
    - Ofrecen un mejor rendimiento que los microkernels y mayor modularidad/seguridad que los monolíticos. Suelen ser una alternativa equilibrada y atractiva.
    - Windows, MacOS



Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **ExoKernel (Kernel de Exoesqueleto):**
    - Provee servicios muy básicos al sistema operativo, evitando alto nivel de abstracción (como hilos o archivos).
    - Ofrece acceso directo a recursos del hardware, permitiendo a los desarrolladores construir soluciones personalizadas
    - Provee primitivas básicas para la gestión de memoria y deja las políticas de gestión de memoria a bibliotecas en espacio de usuario
    - Ofrece primitivas para que las aplicaciones gestionen sus propios procesos.
    - Los controladores se ejecutan en espacio de usuario
    - La mínima cantidad de código reduce los vectores de ataque y los errores, lo que hace que los exokernels sean ideales para entornos especializados como sistemas en tiempo real o embebidos



Kernel – Caracterizaciones

- De acuerdo a su arquitectura de desarrollo, podemos caracterizarlos en distintos tipos:
  - **Kernel de Tiempo Real (Real-Time Kernel):**
    - Debe garantizar que las operaciones se completen dentro de marcos de tiempo conocidos y predecibles.
    - Utiliza algoritmos de scheduling de CPU avanzados para asegurar que las tareas se ejecuten en el orden correcto y cumplir con los plazos establecidos
    - Implementa mecanismos de asignación de memoria estática o dinámica optimizados para prevenir retrasos indeseados, evitando deseablemente los fallos de páginas
    - Soporta multitasking apropiativo donde las tareas de alta prioridad suspenden a las de menor
    - Provee mecanismos de seguridad robustos
    - VxWorks, QNX