1. UNIDAD 1: INTRODUCCION

Escuela de Ingeniería de Ciencias Y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Anuncios Importantes

Tema 1
Tema 2
Tema 3
Tema 4

AGENDA

1. Introducción a los sistemas operativos
2. Introducción a Linux
3. Introducción al kernel de Linux
4. Arquitectura del kernel y estructura de código
5. Escribir y compilar módulos del kernel

COMPETENCIA(S) QUE DESARROLLAREMOS

Entender el marco de referencia o estructura lógica general de un sistema operativo, que le permita la utilización, análisis y diseño de sistemas operativos.

¿QUÉ ES UN SISTEMA OPERATIVO?

El sistema operativo es un sistema que se encarga de manejar todo el software y hardware en una computadora.

La mayor parte del tiempo diferentes programas se ejecutan simultaneamente, cada uno requiriendo acceso a partes de la computadora como el CPU, memoria RAM o almacenamiento.

Es responsabilidad del sistema brindarle a cada uno con los recursos que necesita.

SUPERVISAR EL SISTEMA
Manejar memoria, iniciar y controlar la ejecucion de procesos

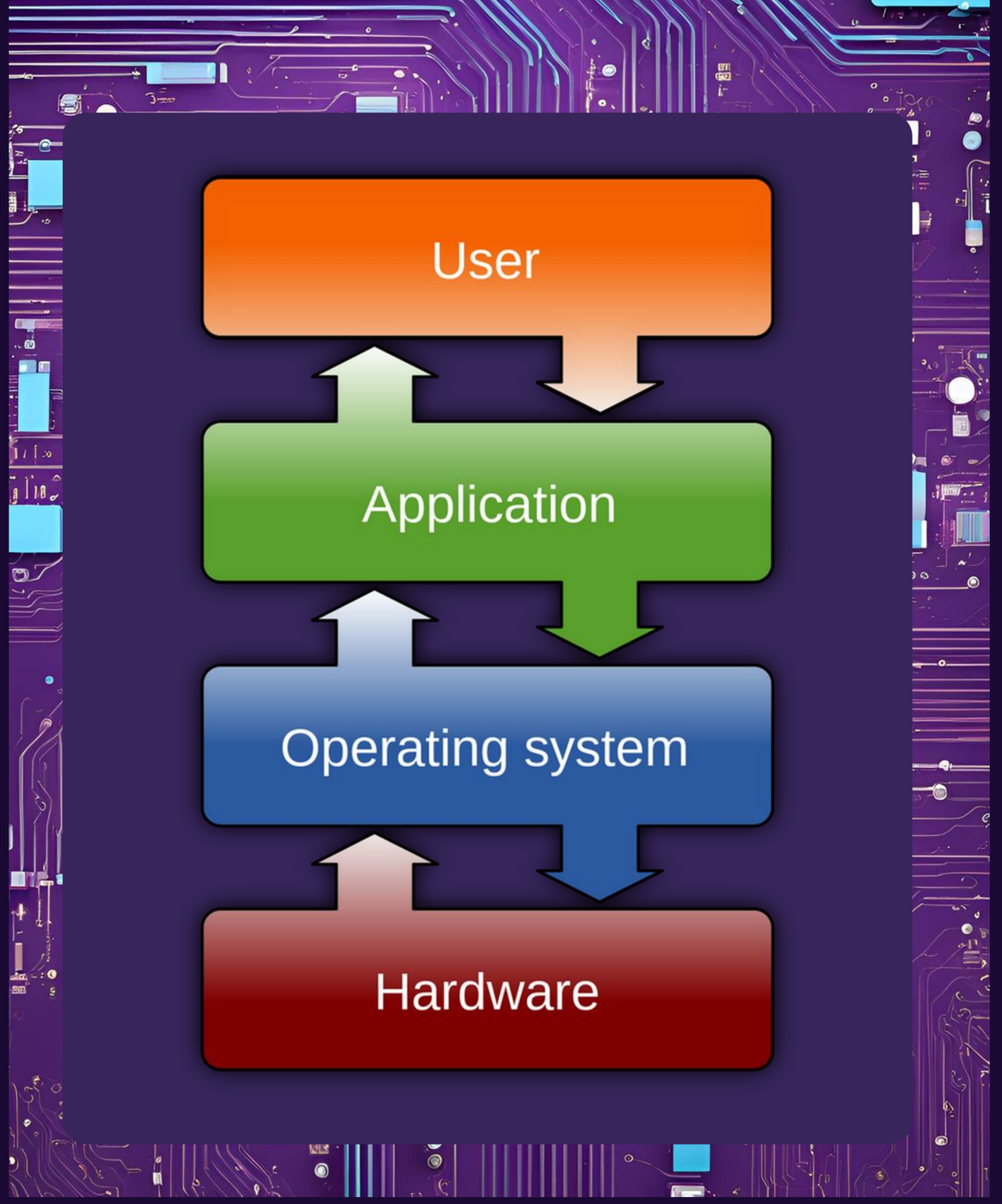
SERVICIOS DE HARDWARE
Proveer los controladores para distintos dispositivos (displays, impresoras, etc)

SERVICIOS DE SOFTWARE
Proveer programas como la interfaz de usuario (UI), sistemas de archivos, etc.

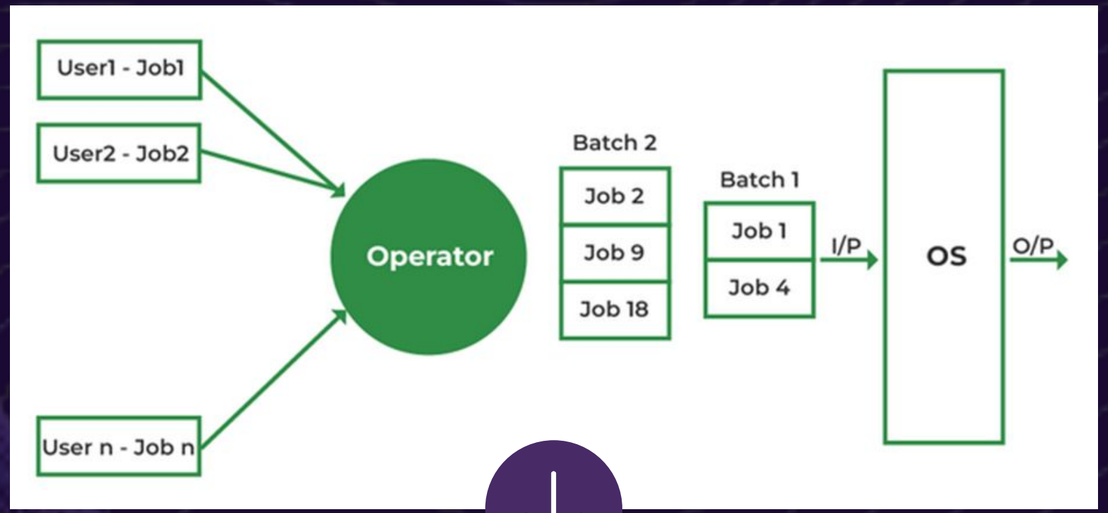
SERVICIOS DE COMUNICACIONES
Redirigir solicitudes de información del sistema local a uno externo

El usuario interactúa con el software de sistema y de aplicación. El software de sistema y de aplicación interactúan con el sistema operativo. El sistema operativo interactúa con el hardware.

Cada una de estas interfaces son transacciones bidireccionales y cada una envía y recibe datos.



TIPOS DE SISTEMAS OPERATIVOS

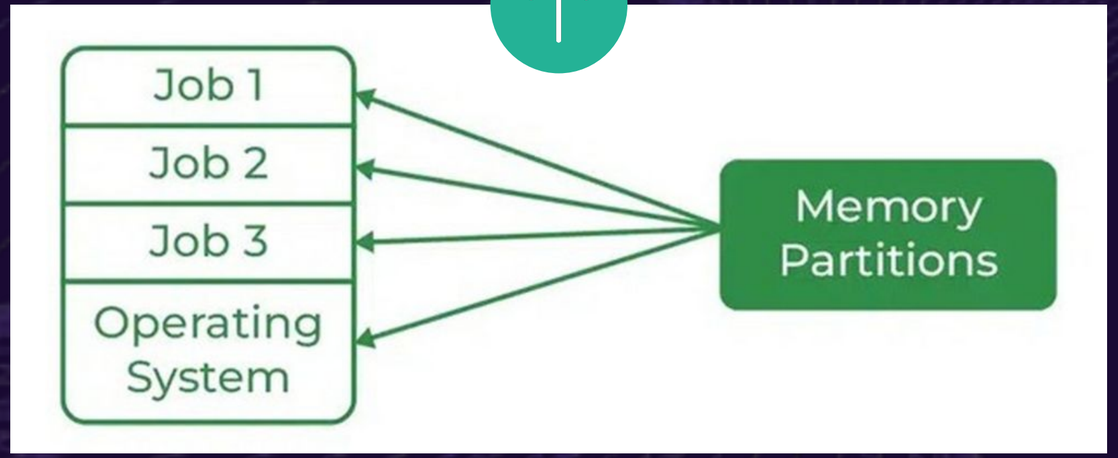


SISTEMA OPERATIVO POR LOTES

El procesamiento por lotes es un tipo de sistema operativo que procesa tipos similares de trabajos en un lote. En el sistema operativo de procesamiento por lotes, varios trabajos de tipos similares se convierten en un lote y se envían a la CPU para su ejecución.

Se puede ilustrar como si más de un programa estuviera presente en la memoria principal y cualquiera de ellos se pudiera mantener en ejecución. Esto se utiliza básicamente para una mejor ejecución de los recursos.

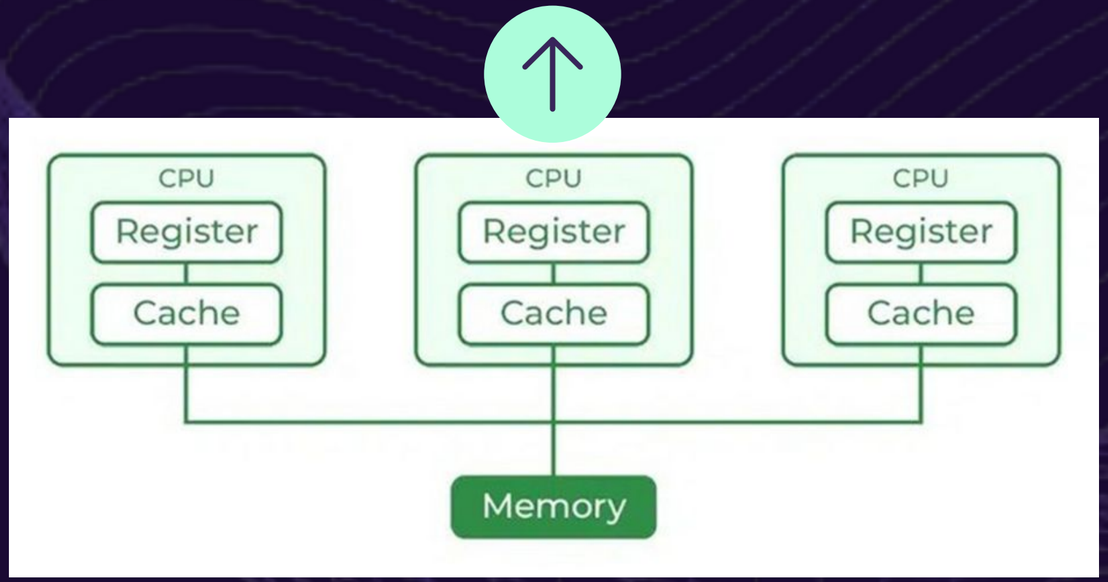
SISTEMA OPERATIVO MULTIPROGRAMACIÓN

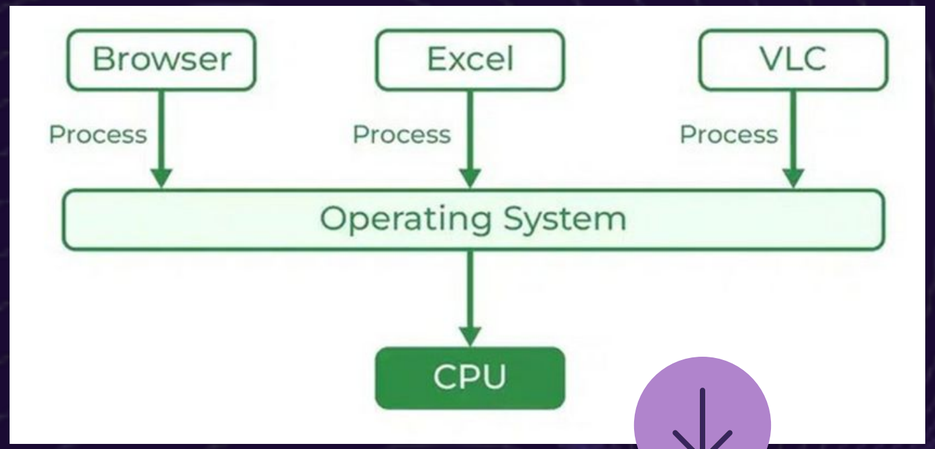


TIPOS DE SISTEMAS OPERATIVOS

Es un tipo de sistema operativo en el que se utiliza más de una CPU para la ejecución de recursos.

SISTEMA OPERATIVO MULTIPROCESAMIENTO

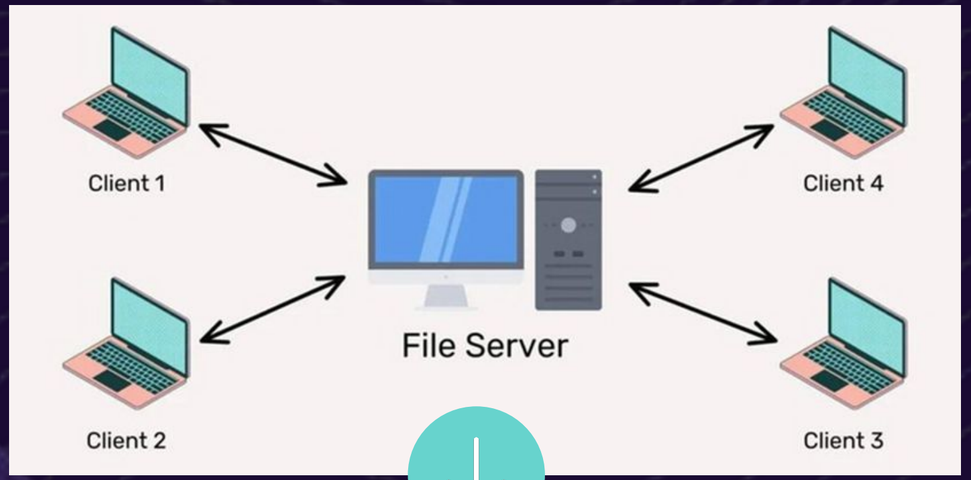




SISTEMA OPERATIVO MULTITAREA

El sistema operativo multitarea es un sistema operativo multiprogramación que cuenta con un algoritmo de planificación Round-Robin que le permite ejecutar múltiples programas simultáneamente.

TIPOS DE SISTEMAS OPERATIVOS

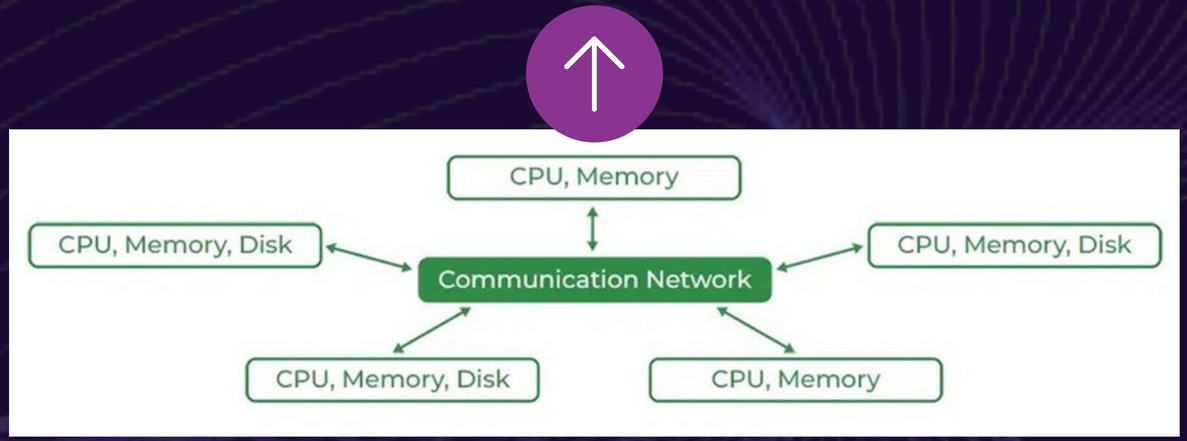


SISTEMA OPERATIVO DE RED

Estos sistemas se ejecutan en un servidor y brindan la capacidad de administrar datos, usuarios, grupos, seguridad, aplicaciones y otras funciones de red. Estos tipos de sistemas operativos permiten el acceso compartido a archivos, impresoras, seguridad, aplicaciones y otras funciones de red a través de una pequeña red privada.

Varias computadoras autónomas interconectadas se comunican entre símediante una red de comunicación compartida. Los sistemas independientes poseensu propia unidad de memoria y CPU.

SISTEMA OPERATIVO DISTRIBUIDO

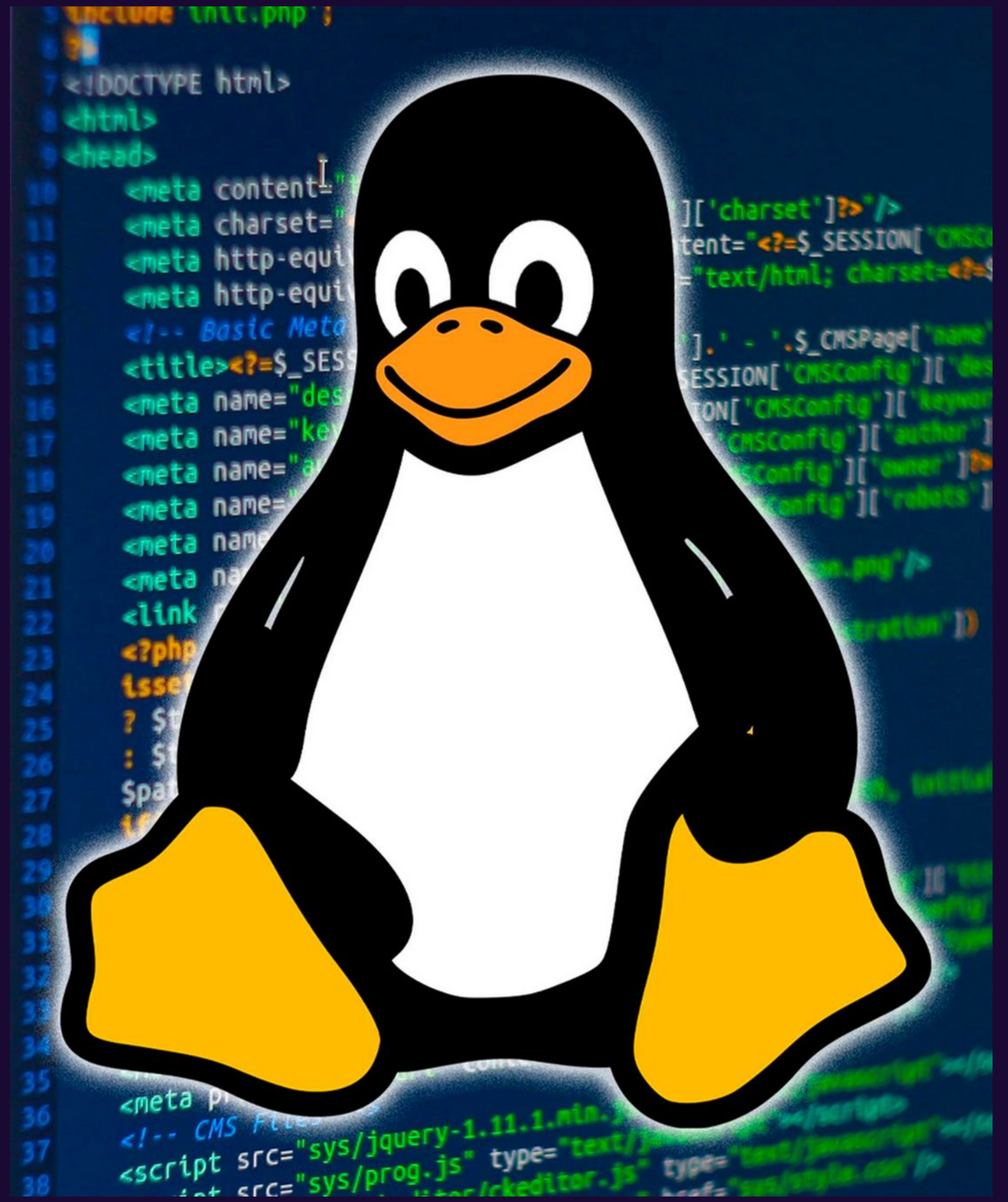


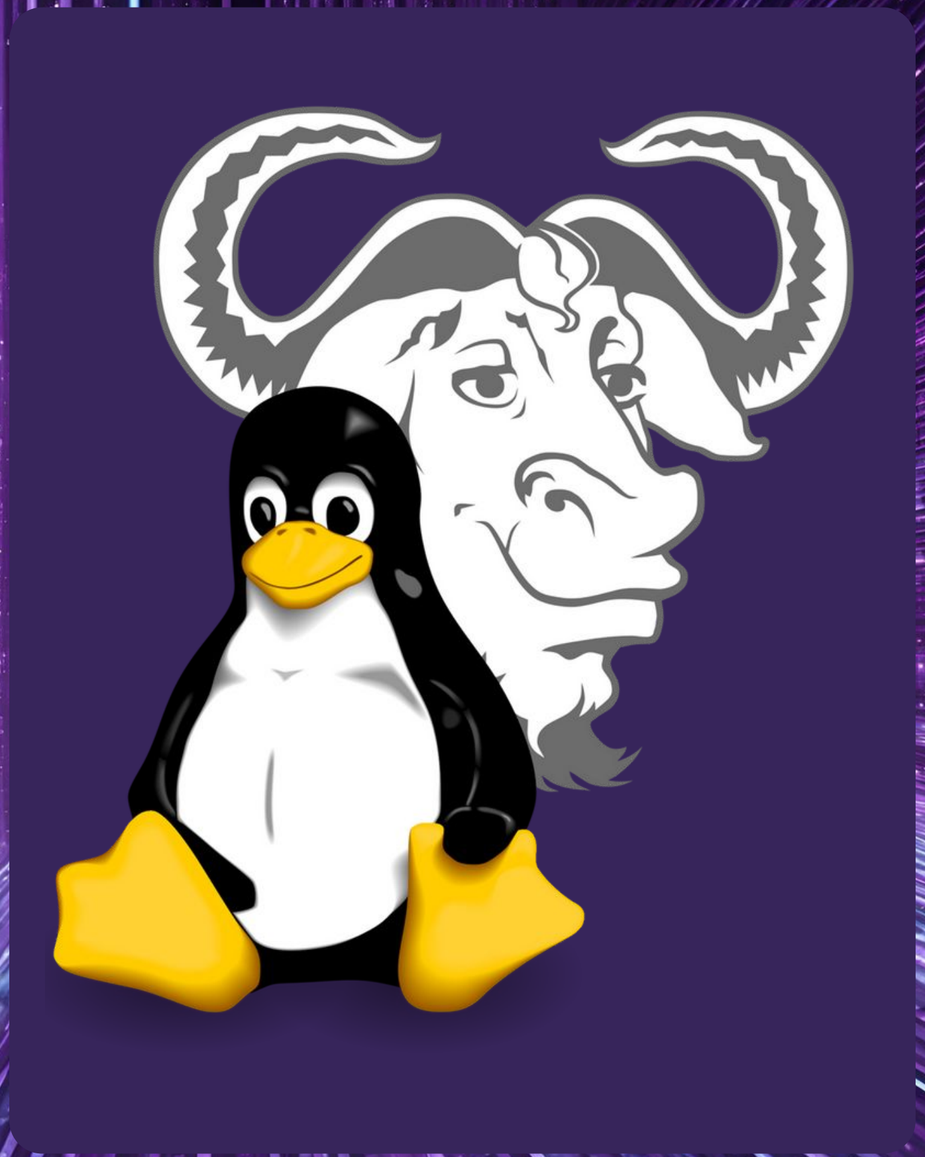
LINUX KERNEL

El kernel de Linux es un kernel de sistema operativo tipo Unix, monolítico, modular, multitarea, gratuito y de código abierto escrito originalmente en 1991 por Linus Torvalds.

Se encuentra dentro del sistema operativo y controla todas las funciones principales del hardware, ya sea un teléfono, una computadora portátil, un servidor o cualquier otro tipo de equipo.

Se puede adaptar a arquitecturas específicas y a varios escenarios de uso utilizando una familia de comandos simples (es decir, sin la necesidad de editar manualmente su código fuente antes de la compilación); los usuarios privilegiados también pueden ajustar los parámetros del kernel en tiempo de ejecución.





GNU/LINUX

Es una familia de sistemas operativos tipo Unix compuesto por software libre y de código abierto.

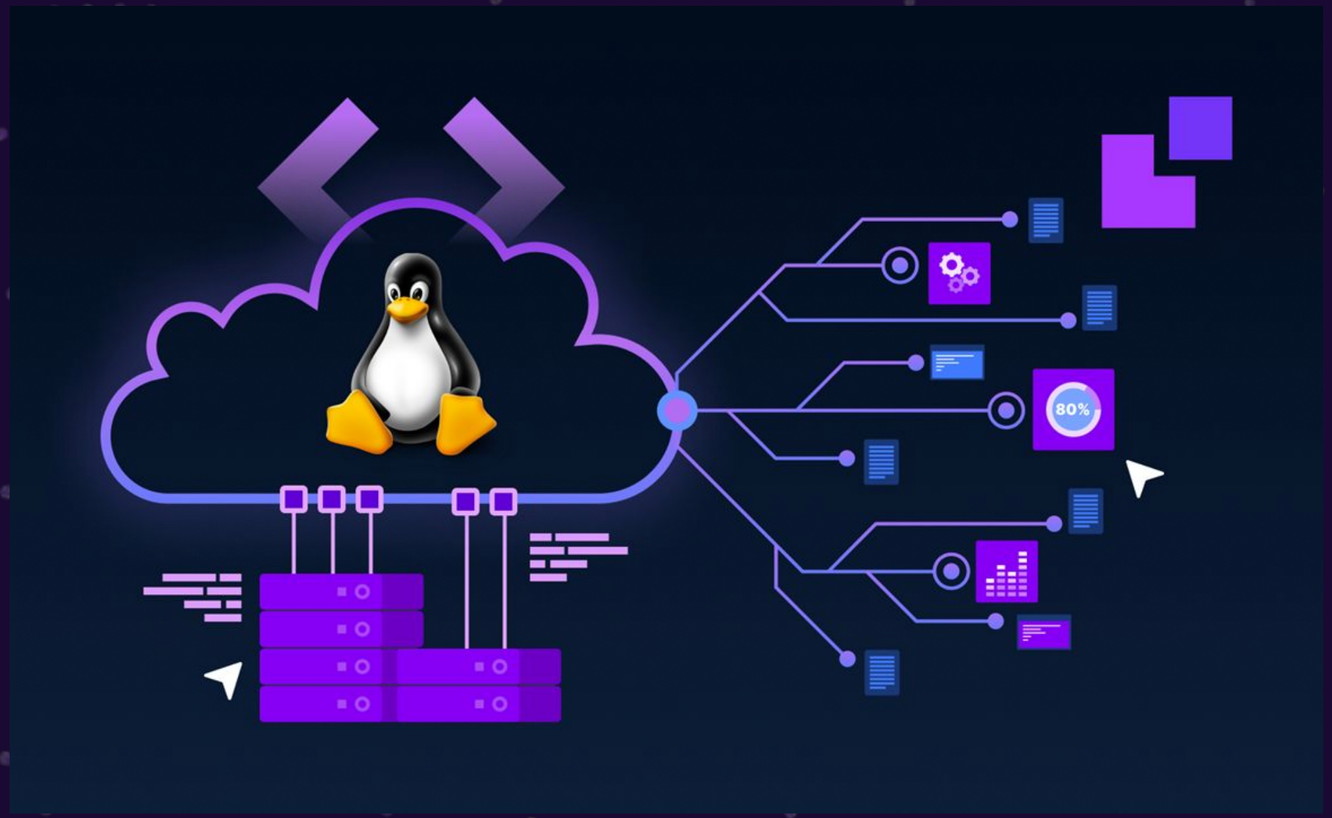
Usualmente se dice ‘Linux’ para referirse al sistema operativo, pero en realidad ese es el nombre del kernel, que representa menos del 50 por ciento de todo el código del sistema. El sistema completo está formado también por una gran cantidad de componentes del Proyecto GNU junto a componentes de terceros, que van desde compiladores hasta entornos de escritorio.

Cabe señalar que existen derivados que usan el núcleo Linux pero que no tienen componentes GNU, como el sistema operativo Android, asi como también existen distribuciones de software GNU donde el núcleo Linux está ausente como FreeBSD.

¿POR QUE USAR LINUX?

Actualmente, Linux se utiliza para gestionar múltiples servicios, como programación de procesos, programación de aplicaciones, dispositivos periféricos fundamentales y sistemas de archivos. Linux puede ejecutar cualquier tipo de programa que esté acostumbrado a ejecutar en Windows o macOS, desde calculadoras básicas hasta sofisticadas herramientas de edición de gráficos.

Kubernetes, Docker y Open Daylight (utilizados para acelerar la adopción virtualización de funciones de red) son ejemplos de productos basados en Linux que han tenido un impacto significativo en la industria de TI y se han vuelto indispensables para las pilas tecnológicas de una gran cantidad de empresas. organizaciones. Además, los proveedores de servicios en la nube prefieren los sistemas operativos basados en Linux debido a su naturaleza gratuita y de código abierto.



CONTEXTOS QUE UTILIZAN LINUX

01

SISTEMAS SIN CABEZA

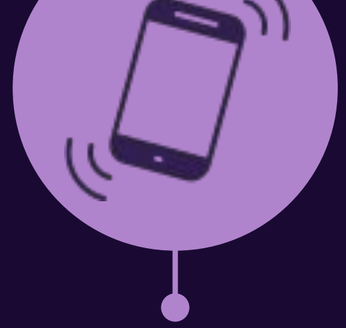
Linux se utiliza para sistemas de servidores sin cabeza que no necesitan una interfaz gráfica de usuario (GUI) o una terminal y un teclado directamente vinculados. Los sistemas sin cabeza se utilizan a menudo para servidores de red y otros dispositivos operados de forma remota

02

DISPOSITIVOS DE RED

Linux es un sistema operativo de red común para enrutadores, conmutadores, servidores DNS, dispositivos de red domésticos y más. Cisco, por ejemplo, proporciona una versión del sistema operativo Cisco Internetwork (IOS) que emplea el kernel de Linux.

03



DISPOSITIVOS INTEGRADOS

Linux se utiliza como sistema operativo integrado para una variedad de aplicaciones, como electrodomésticos, sistemas de entretenimiento para automóviles y dispositivos de sistemas de archivos en red con necesidades informáticas modestas.

04

SERVIDORES

Linux es adecuado para todo tipo de aplicaciones de servidor debido a su capacidad para ejecutar aplicaciones de gran volumen y subprocesos múltiples.

KERNEL LINUX

El kernel de Linux es el núcleo esencial de los sistemas operativos Linux y actúa como la interfaz principal entre el hardware de una computadora y sus procesos. Su función es permitir la comunicación eficiente entre ambos y gestionar de forma óptima los recursos del sistema.

Llamado "kernel" debido a su posición interna dentro del sistema operativo —similar a una semilla en el centro de una fruta de cáscara dura—, este componente controla todas las funciones esenciales del hardware, ya sea en teléfonos, computadoras portátiles, servidores u otros dispositivos. Como corazón del sistema, el kernel asegura que cada componente del hardware responda adecuadamente a las necesidades de los procesos y aplicaciones, optimizando el rendimiento y la estabilidad del equipo.

La arquitectura del kernel de Linux sigue un modelo de capas en el que cada capa tiene responsabilidades específicas y comunica con las capas adyacentes

- **Gestor de Procesos** Administra la creación, planificación y finalización de procesos. Implementa algoritmos para la multitarea y optimiza el uso de la CPU.

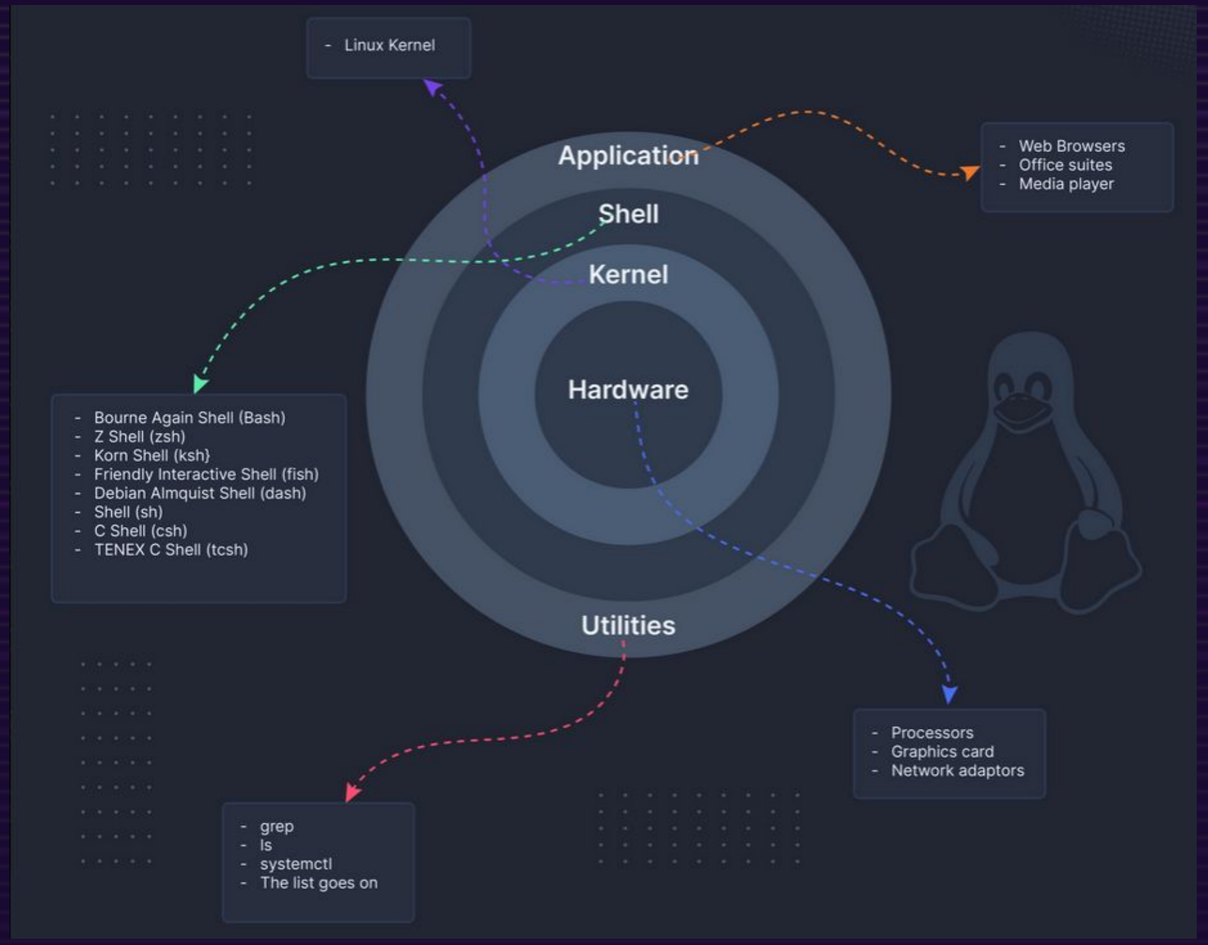
- **Gestión de Memoria** Controla la asignación y organización de la memoria del sistema (virtual y física) mediante paginación y segmentación, asegurando un uso eficiente entre procesos.

SUBSISTEMA DE ARCHIVOS

Ofrece una interfaz común para manipular archivos y soporta múltiples sistemas de archivos (ext4, FAT32, NTFS), facilitando la gestión de datos en distintos medios de almacenamiento.

- **Redes y Protocolo** Maneja la comunicación de red, soportando protocolos como TCP/IP y UDP para la transmisión de datos y conexión a Internet.

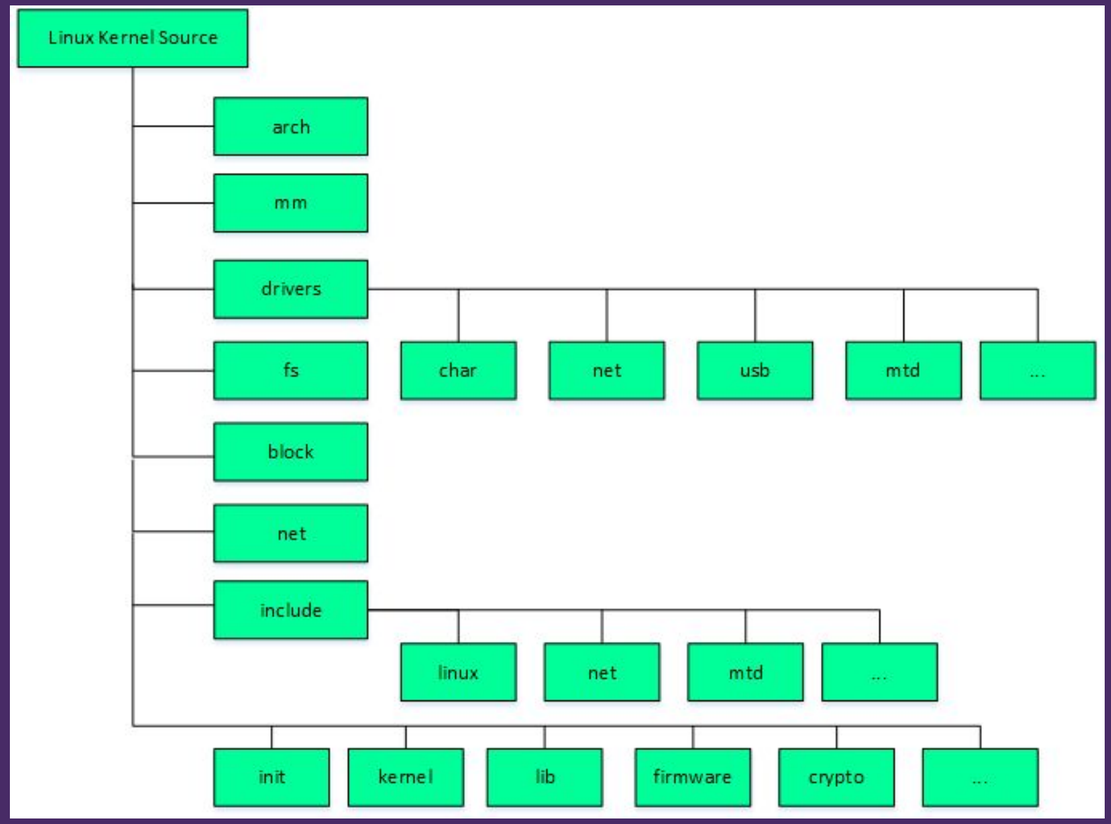
- **Interfaz de Llamadas al Sistema (Syscalls)** Proporciona una API para que las aplicaciones soliciten servicios del kernel, como acceso a archivos, operaciones de red y gestión de memoria.



Imagina que el kernel es el asistente personal ocupado de un alto ejecutivo, que en este caso es el hardware. Su tarea es transmitir mensajes y solicitudes (procesos) de los empleados y del público (usuarios) al ejecutivo, recordar qué elementos están almacenados y en qué lugar (memoria), y decidir quién puede acceder al ejecutivo en cada momento y por cuánto tiempo.

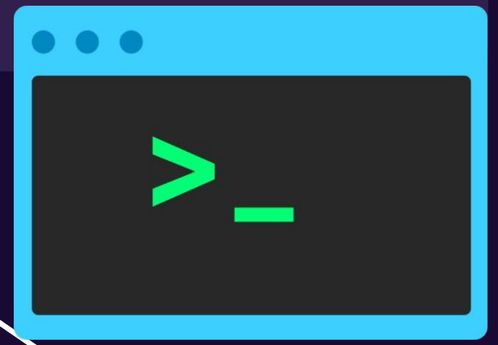
ESTRUCTURA DEL CÓDIGO DEL KERNEL DE LINUX

El código fuente del kernel de Linux está organizado en varios subdirectorios dentro del árbol de código principal, cada uno dedicado a un subsistema específico. Esta estructura modular permite mantener el código bien organizado y facilita el mantenimiento y desarrollo de nuevas características. Los principales directorios y archivos son:



- **/arch:** Contiene el código específico para diferentes arquitecturas de hardware (como x86, ARM, MIPS, etc.). Aquí se encuentran las configuraciones y adaptaciones del kernel para funcionar en diferentes plataformas.
- **/kernel:** Incluye el núcleo del código del sistema operativo, con funcionalidades como la gestión de procesos, sincronización, y planificación. Es el corazón del sistema y maneja la lógica central del kernel.
- **/include**: Contiene los archivos de cabecera utilizados por el resto del kernel. Estos archivos definen estructuras de datos, funciones, y constantes usadas a lo largo del código.
- **/fs:** Código relacionado con los sistemas de archivos soportados por Linux. Aquí se implementan los diferentes tipos de sistemas de archivos y el VFS (Virtual File System).
- **/drivers:** Contiene los controladores de dispositivos para el hardware, como controladores de red, de almacenamiento, de video, y otros periféricos. Esta estructura modular permite que los drivers se carguen y descarguen dinámicamente, mejorando la adaptabilidad del kernel.

- **/net:** Incluye el código para la pila de redes y los protocolos de comunicación, como TCP/IP, IPv4, IPv6, y otros. Este directorio es fundamental para todas las funcionalidades de red del kernel.
- **/mm:** Contiene el código de gestión de memoria, incluyendo algoritmos de administración de memoria virtual, paginación, intercambio (swapping) y asignación de memoria. Esta sección es crucial para optimizar el uso de recursos y evitar cuellos de botella.
- **/security:** Código relacionado con la seguridad del sistema, como el módulo SELinux, AppArmor, y otros mecanismos de control de acceso. También se ocupa de la implementación de políticas de seguridad para proteger el sistema de amenazas.
- **/init:** Contiene el código de inicialización del kernel que se ejecuta al inicio del sistema. Este código prepara el entorno y carga las configuraciones necesarias para que el kernel pueda iniciar correctamente.



MODULOS DE KERNEL

Los módulos del kernel son componentes de código que se pueden cargar y descargar en tiempo de ejecución, lo que permite extender las funcionalidades del kernel sin necesidad de recompilar el sistema. Esto proporciona flexibilidad y facilidad de mantenimiento al sistema operativo.

ESPACIO DE KERNEL VS. ESPACIO DEL USUARIO

ESPACIO DEL KERNEL

Es el entorno donde se ejecuta el código del kernel, que tiene acceso completo al hardware. Este espacio es privilegiado y permite la gestión directa de los recursos del sistema.

ESPACIO DE USUARIO

Aquí se ejecutan las aplicaciones de usuario, las cuales tienen un acceso limitado al hardware. Esta restricción garantiza la seguridad y estabilidad del sistema, impidiendo que las aplicaciones interfieran directamente con el funcionamiento del kernel y de otros procesos del sistema.

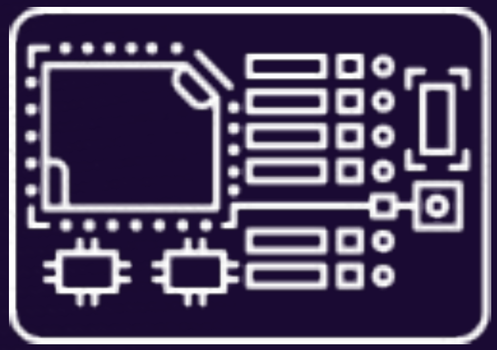
ESCRIBIR UN MODULO DE KERNEL

Usando de base un archivo *hello.c*

```c
// hello.c
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("<NombreModulo>");
MODULE_DESCRIPTION("Un módulo simple que imprime mensajes.");
MODULE_VERSION("0.1");

static int __init hello_init(void) {
    printk(KERN_INFO "Hola, mundo! El módulo ha sido cargado.\n");
    return 0; // Devuelve 0 si se carga correctamente
}

static void __exit hello_exit(void) {
    printk(KERN_INFO "Adiós, mundo! El módulo ha sido descargado.\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Se necesita cear un archivo *Makefile*

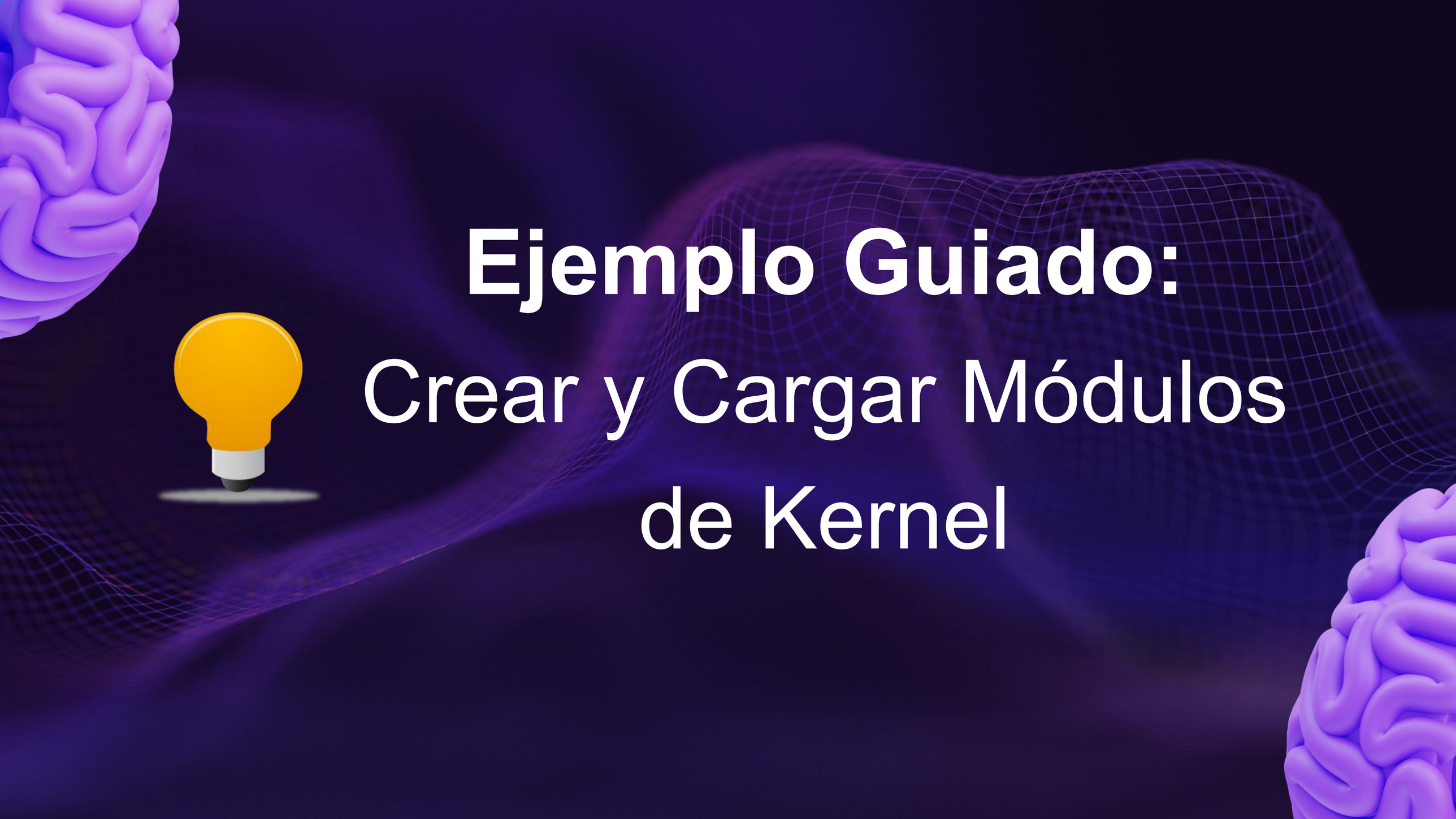
```
# Makefile
obj-m += hello.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```



COMPILAR UN MODULO DE KERNEL

Para compilar el nuevo modulo se necesita abrir una terminal y navegar al directorio donde se alojan los archivos **hello.c y Makefile**. Luego, ejecutar los siguientes comando para compilar el módulo

```
> Compilar el modulo
$ make

> Cargar el modulo
$ sudo insmod hello.ko

> Verificar el Mensaje del Kernel
$ dmesg | tail

> Descargar el modulo
$ sudo rmmod hello.ko

> Verificar el Mensaje del Kernel
$ dmesg | tail
```


Ejemplo Guiado:
Crear y Cargar Módulos
de Kernel

CONCEPTOS CLAVE APRENDIDOS

- Que es un sistema operativo
- Tipos de Sistemas
- GNU/Linux
- Kernel de Linux
- Arquitectura Esencial de un SO

VALOR DE LA SEMANA

- **Responsabilidad:** en contexto de programas libres como linux, se trata de código fuente de un sistema crítico como linux, por lo que las modificaciones deben ser responsables y nunca con fines negativos.

¡GRACIAS POR LA ATENCIÓN!

¿Dudas?