



LABORATORIO SISTEMAS OPERATIVOS 2

# INTRODUCCION AL KERNEL DE LINUX

# 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.

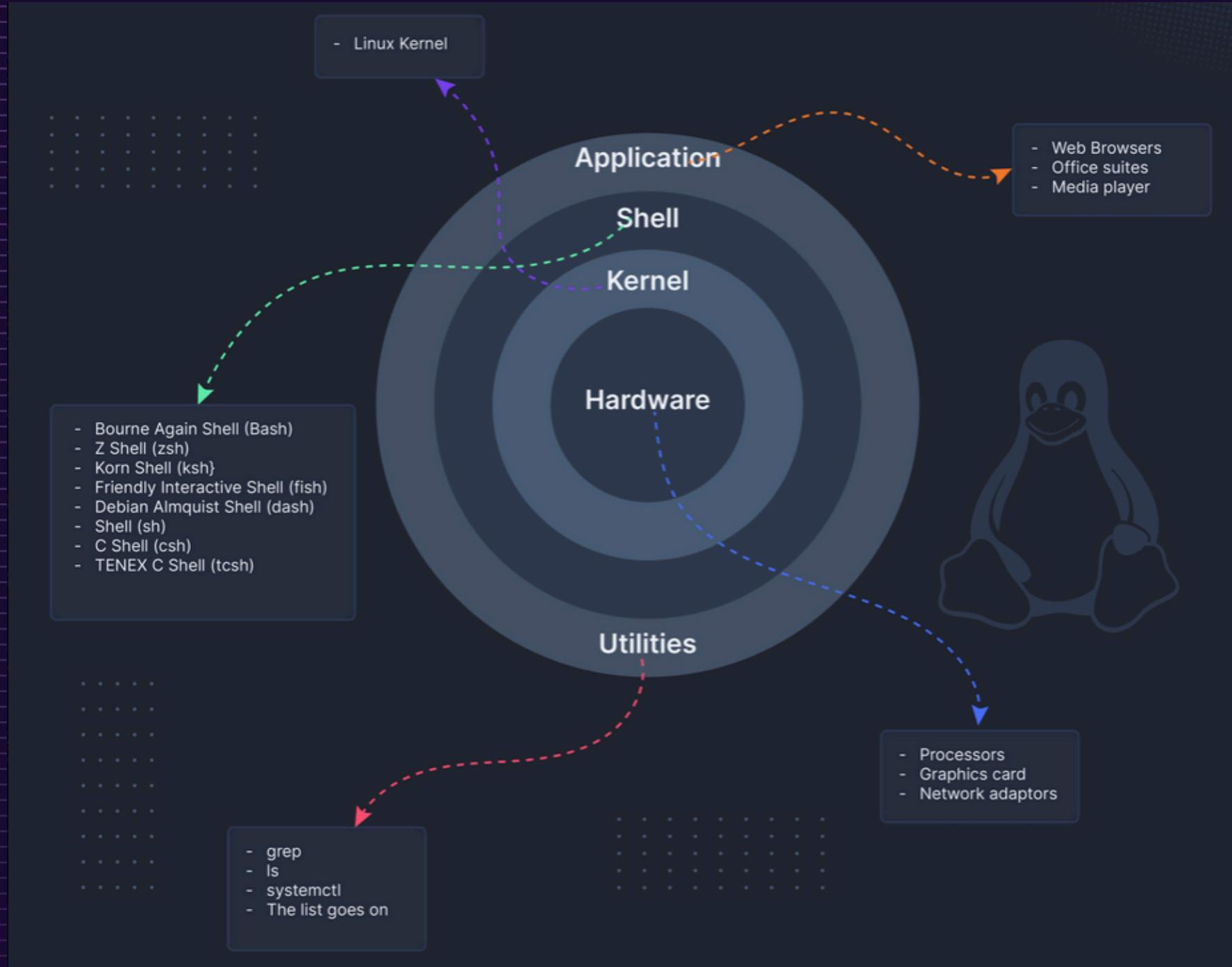


### SUBSISTEMA DE ARCHIVOS

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.



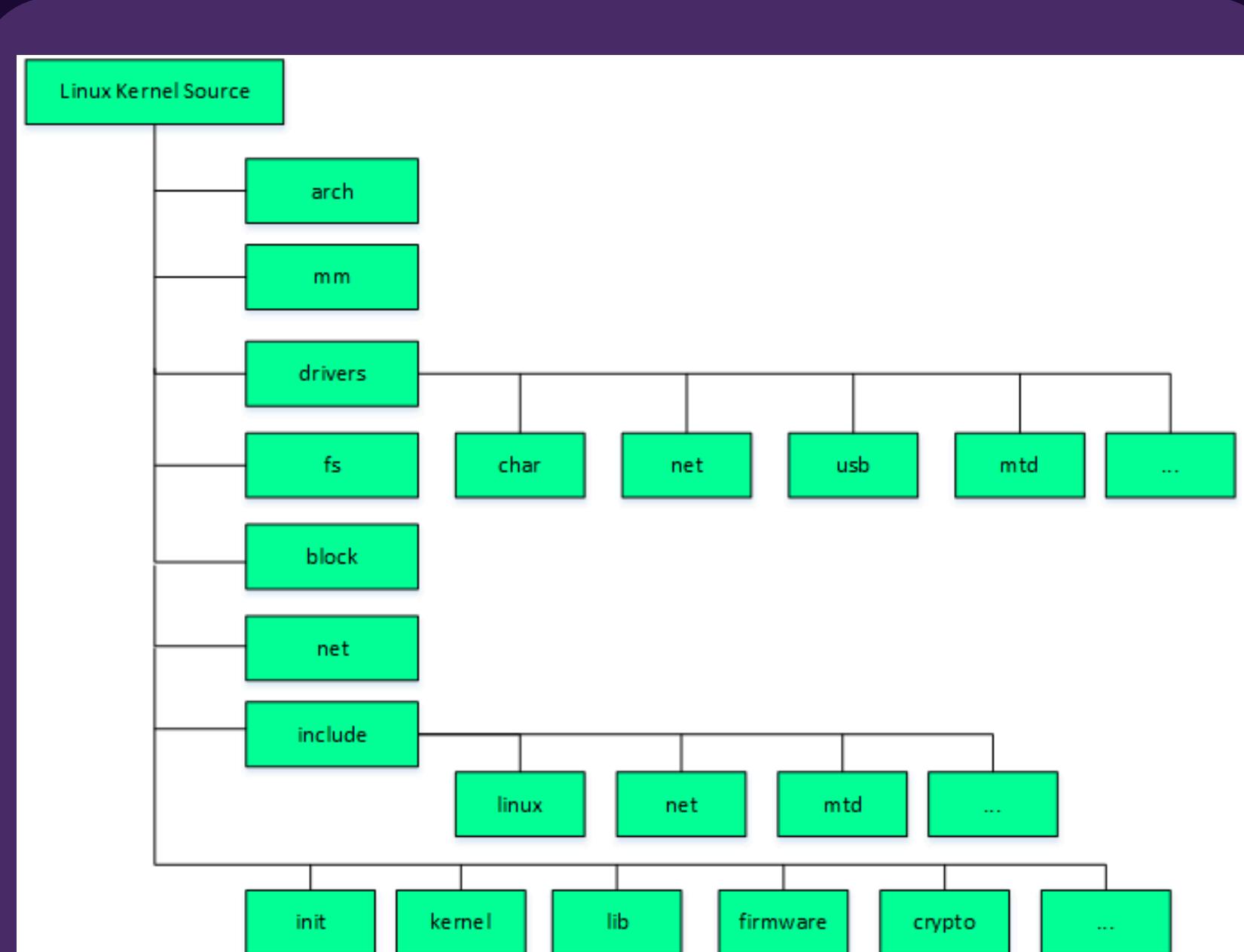
- **Redes y Protocolo** Maneja la comunicación de red, soportando protocolos como TCP/IP y UDP para la transmisión de datos y conexión a Internet.
- **Interfaz de Llamadas al Sistema (Syscalls)** Proporciona una API para que las aplicaciones soliciten servicios del kernel, como acceso a archivos, operaciones 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.

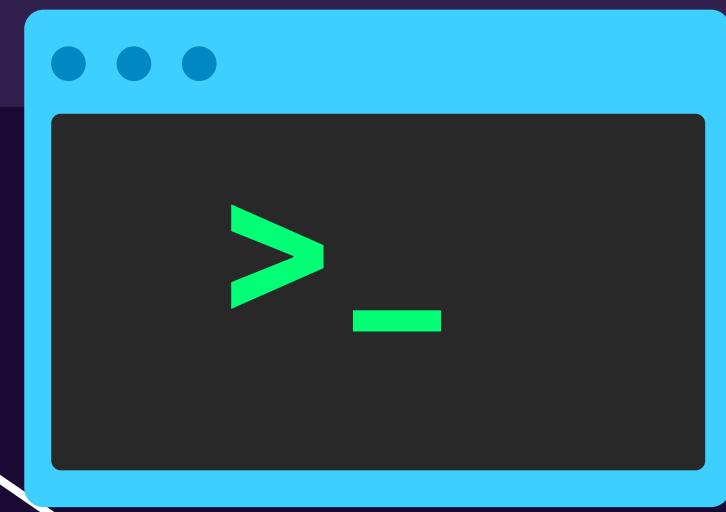
# ESTRUCTURA DEL CÓDIGO DEL KERNEL DE LINUX

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:



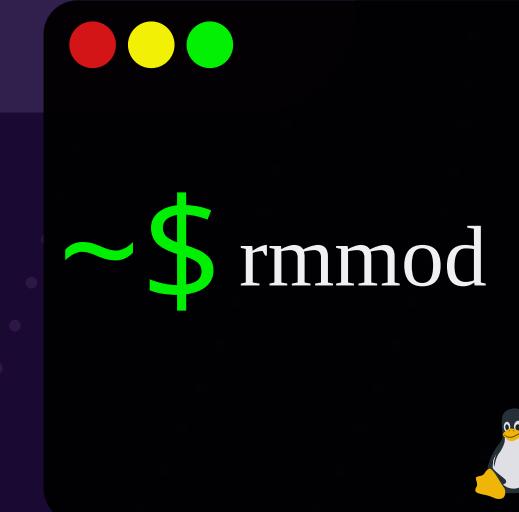
- **/arch:** Contiene el código específico para diferentes arquitecturas de hardware (como `x86`, ARM, MIPS, etc.). Aquí se encuentran las configuraciones y adaptaciones del kernel para funcionar en diferentes plataformas.
- **/kernel:** Incluye el núcleo del código del sistema operativo, con funcionalidades 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.

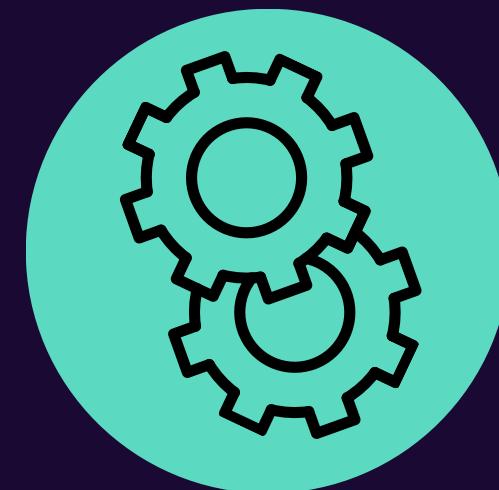


# MODULOS DE KERNEL

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](#)

```
// 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");
}

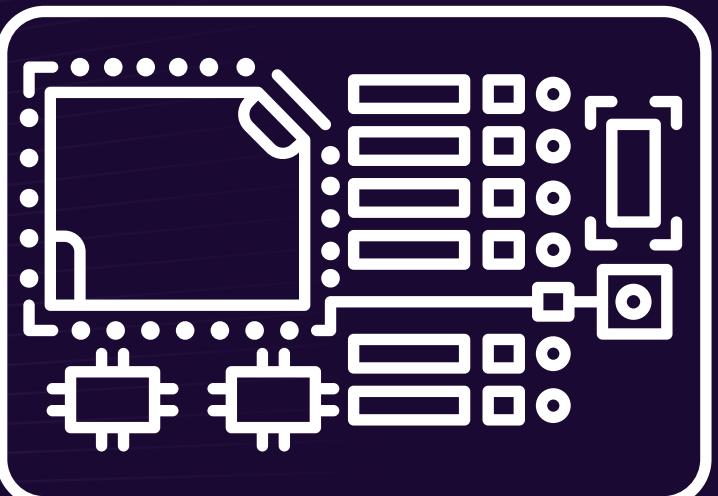
module_init(hello_init);
module_exit(hello_exit);
```

Se necesita crear un archivo [Makefile](#)

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

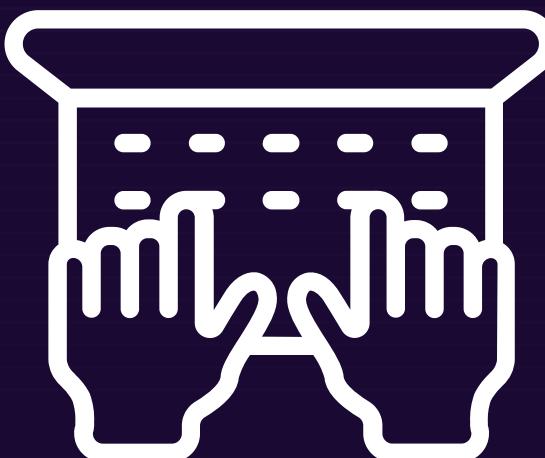
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

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`

# MODIFICANDO EL KERNEL

## Obtener el Código Fuente del Kernel

- Comenzar descargando la versión más reciente del código fuente del kernel desde [kernel.org](https://kernel.org). Esto se puede hacer mediante comandos de terminal o desde el navegador. Una vez descargado, se debe descomprimir el archivo para acceder a los archivos de código fuente.

## Instalar Dependencias Necesarias:

- Antes de compilar el kernel, es esencial contar con todas las herramientas necesarias, incluyendo compiladores y bibliotecas



> Instalar dependencias

```
$ sudo apt-get install build-essential libncurses-dev bison flex libssl-dev libelf-dev
```

The screenshot shows the homepage of The Linux Kernel Archives. At the top, there's a navigation bar with links for About, Contact us, FAQ, Releases, Signatures, and Site news. To the right of the navigation is a small Tux the Penguin icon. Below the navigation, a large button on the right side displays "Latest Release 6.11.6" with a download icon. To the left of this button, there's a table showing download links for different protocols: HTTP, GIT, and RSYNC. The table lists various kernel versions with their respective dates and download links. The main content area below the table lists kernel releases in a table format, showing columns for name, date, tarball, PGP, patch, inc. patch, view diff, browse, and changelog.

Protocol	Location
HTTP	<a href="https://www.kernel.org/pub/">https://www.kernel.org/pub/</a>
GIT	<a href="https://git.kernel.org/">https://git.kernel.org/</a>
RSYNC	<a href="rsync://rsync.kernel.org/pub/">rsync://rsync.kernel.org/pub/</a>

Release	Date	Tarball	PGP	Patch	Inc. Patch	View Diff	Browse	Changelog
mainline: 6.12-rc5	2024-10-27	[tarball]		[patch]	[inc. patch]	[view diff]	[browse]	
stable: 6.11.6	2024-11-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable: 6.10.14 [EOL]	2024-10-10	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 6.6.59	2024-11-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 6.1.115	2024-11-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 5.15.170	2024-11-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 5.10.228	2024-10-22	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 5.4.284	2024-09-12	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm: 4.19.322	2024-09-12	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next: next-20241101	2024-11-01							

## Configurar el Kernel:

Se gestiona a través de un archivo .config. Para facilitar este proceso, se puede utilizar el comando `localmodconfig`, que ajusta automáticamente el archivo para incluir solo los módulos del kernel actualmente cargados en el sistema. Sin embargo, es importante destacar que esta configuración hará que el kernel compilado sea específico para la máquina utilizada.

## Compilar el Kernel:

Usar el comando:

```
$ fakeroot make
```

**fakeroot** es una herramienta esencial que permite ejecutar comandos en un entorno simulado con permisos de superusuario, sin necesidad de tener realmente dichos permisos. Esto es particularmente útil al usar make, ya que facilita la creación de archivos con las propiedades y permisos de superusuario, simplificando así el proceso de construcción y empaquetado de software.

## Instalar el Kernel:

Primero se instalan los módulos del kernel ejecutando:

```
$ sudo make modules_install
```

Luego instalamos el kernel:

```
$ sudo make install
```

Después de eso, reiniciamos la computadora para que se complete la instalación.



> Configurar el kernel

```
$ make Localmodconfig
```

> Deshabilitar llaves privadas

```
$ scripts/config --disable SYSTEM_TRUSTED_KEYS
```

```
$ scripts/config --disable SYSTEM_REVOCATION_KEYS
```

```
$ scripts/config --set-str CONFIG_SYSTEM_TRUSTED_KEYS ""
```

```
$ scripts/config --set-str CONFIG_SYSTEM_REVOCATION_KEYS ""
```

> Compilar el kernel

```
$ fakeroot make -j4
```

> Instalar el kernel

```
$ sudo make modules_install
```

```
$ sudo make install
```

```
$ sudo reboot
```





**¡GRACIAS POR  
LA ATENCIÓN!**

**¿Dudas?**