

MÓDULO ISO

Introducción a Linux

Capítulo 1 — Historia, Kernel, Distribuciones y Comandos

1º ASIR · IES Aljada, Puente Tocinos (Murcia)
Administración de Sistemas Informáticos en Red



Linux® — Linus Torvalds, 1991

"El software que impulsa el mundo"

1.1 — ¿Qué es Linux?



KERNEL

Linux es, en sentido estricto, el kernel: el núcleo central que gestiona hardware, memoria, procesos y dispositivos. Es la capa más baja del SO.



DISTRIBUCIÓN

«Linux» coloquialmente = Kernel + herramientas GNU + aplicaciones de usuario, empaquetadas en una distribución instalable.



CÓDIGO ABIERTO

El código fuente es libre: cualquiera puede leerlo, modificarlo y redistribuirlo bajo la licencia GNU GPL. Desarrollado por la comunidad.

1.2.1 — El Rol del Kernel

Gestión de memoria

Asigna y libera RAM entre procesos. Crea la ilusión de memoria continua.

Planificación de procesos

Multitarea preferente: decide qué proceso usa la CPU en cada instante.

Gestión de E/S

Abstrae el acceso a disco, red y periféricos mediante drivers y syscalls.

Arranque del sistema

El gestor de arranque (GRUB) carga el kernel, que inicia el espacio de usuario.



1.2.3 — El Rol del Código Abierto y el Proyecto GNU

```
# Linux Kernel 1991
void kernel_init() {
    memory_init();
    process_init();
    filesystem_init();
}
```

```
// GPL License - Free to use,
// modify and distribute!
```

CÓDIGO ABIERTO
GNU GPL License



Linus Torvalds, 1991

Código Fuente Disponible

Tienes derecho a obtener el código fuente, leerlo, modificarlo y redistribuirlo. Es la base filosófica del movimiento.

Linus Torvalds, 1991

Estudiante finlandés que liberó el núcleo Linux como hobby. La comunidad lo adoptó rápidamente y lo expandió.

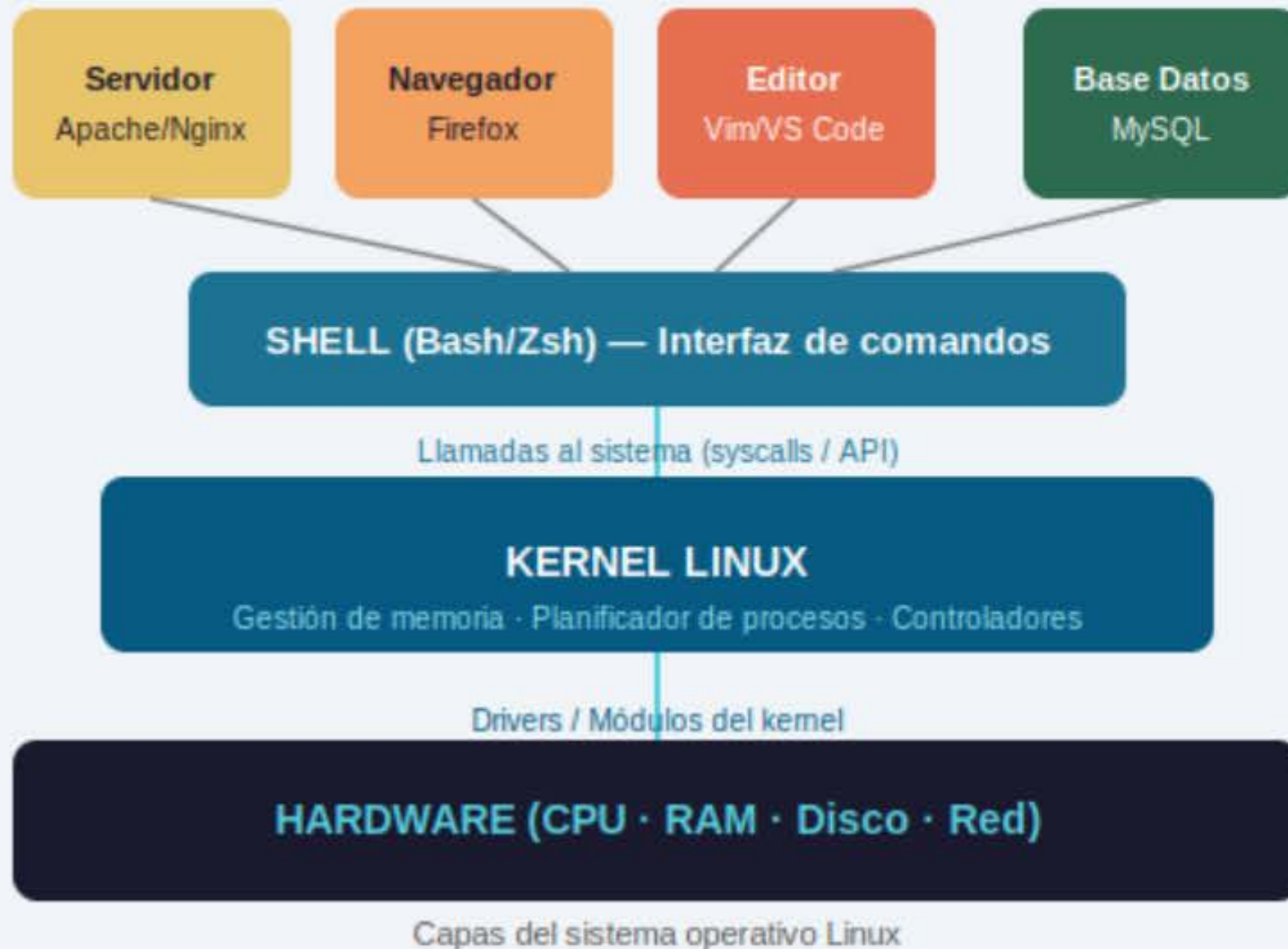
GNU Proyecto GNU (Richard Stallman)

Aportó compiladores (GCC), shell (Bash), utilidades coreutils y la licencia GPL. Linux + GNU = sistema completo.

Licencia GPL

Copyleft: cualquier derivado debe distribuirse también con código fuente y bajo GPL. Garantiza libertades perpetuas.

1.2.2 — Aplicaciones, Procesos y Capas del Sistema



Conceptos Clave

Proceso

Instancia de un programa en ejecución, gestionada por el kernel. Una app puede generar múltiples procesos.

API / Syscall

Interfaz que permite a las apps solicitar servicios al kernel (abrir ficheros, crear sockets, asignar memoria...).

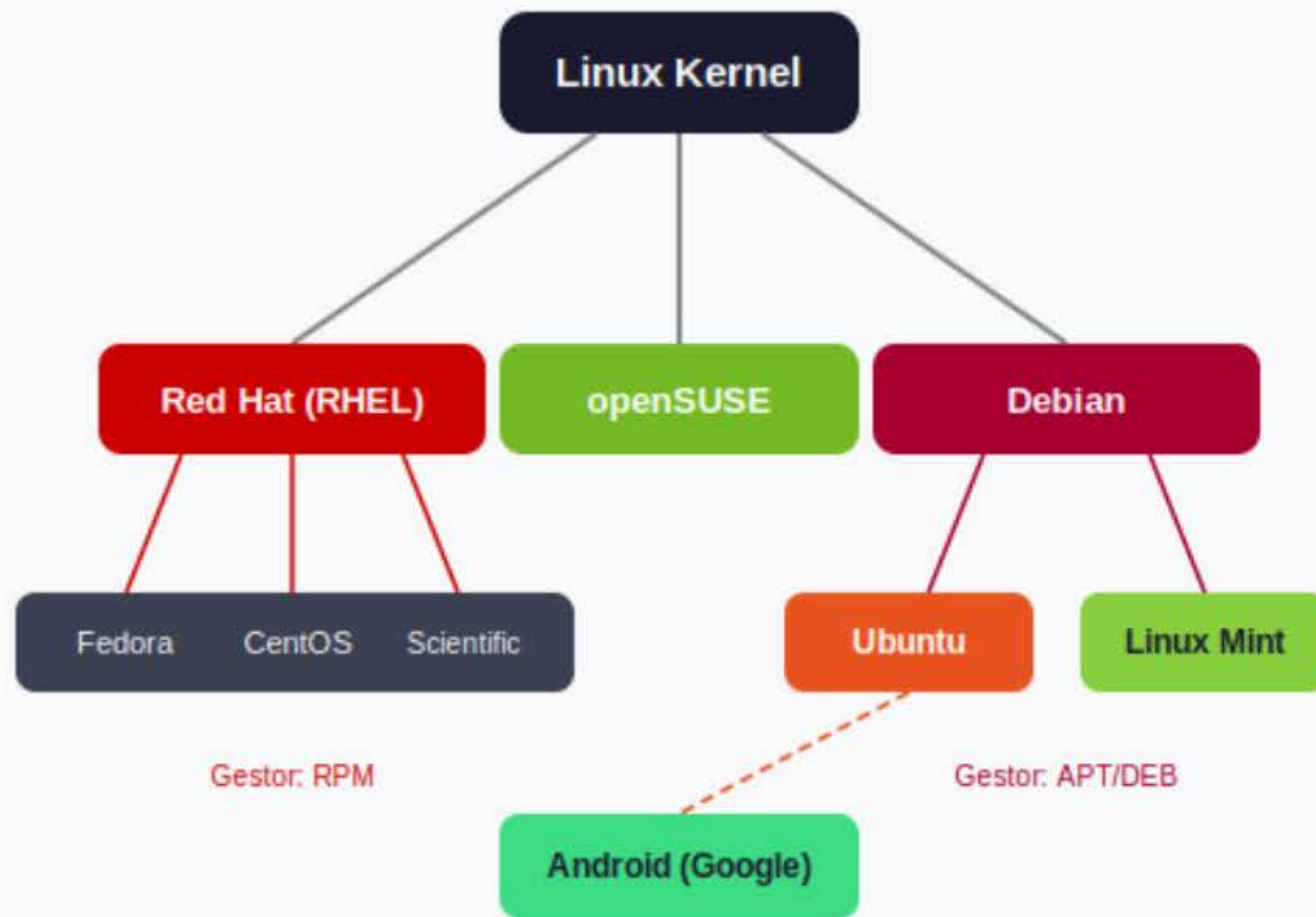
Abstracción

Las apps no necesitan conocer el hardware real. El kernel presenta una interfaz uniforme independiente del dispositivo.

Driver

Módulo del kernel que permite controlar un dispositivo hardware específico.

1.2.4 — Distribuciones de Linux



Árbol de Distribuciones Linux

Familia Red Hat

RHEL: Ciclo largo (~10 años). Soporte comercial. Entornos empresariales.

Fedora: Ciclo corto (~6 meses). Comunidad. Banco de pruebas de RHEL.

CentOS: Recompilación gratuita de RHEL. Sin soporte oficial de Red Hat.

Gestor: RPM — Red Hat Package Manager

Familia Debian

Debian: Base de muchas distros. Fuerte enfoque en software libre.

Ubuntu: Distro derivada. Canonical. Ciclos LTS (5 años) y estándar.

Linux Mint: Derivada de Ubuntu. La más popular para escritorio.

Gestor: APT / DEB — Advanced Package Tool

1.2.4.1 — ¿Qué es un Comando? Tipos

01 Builtin del shell
`cd, echo, pwd, export`
Integrados en Bash. No requieren proceso externo. Se ejecutan dentro del propio shell.

02 Comandos externos
`ls, cp, mv, grep, find`
Ejecutables en el sistema de archivos. El shell los localiza recorriendo el PATH.

03 Alias
`alias ll='ls -la'`
Apodos definidos por el usuario. Sustituyen a un comando o serie de comandos. Se definen en `~/.bashrc`.

04 Funciones
`function mi_func() { ... }`
Bloques de código reutilizables dentro del shell. Más potentes que los alias.



```
Terminal — bash

user@linux:~$ cd /home
+ Builtin del shell (bash interno)

user@linux:~$ ls -la
+ Comando externo (PATH: /bin/ls)

user@linux:~$ alias ll='ls -la'
+ Alias personalizado

user@linux:~$ my_func() { echo "Hola"; }
+ Función de shell
```

`PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin`

El shell busca el ejecutable en cada directorio del PATH

4 tipos de comandos en Bash

1.2.5 — Plataformas de Hardware

Linux está en todas partes

Supercomputadores

96.4% del Top500 mundial ejecuta Linux

Servidores web

~72% del mercado global de servidores

Smartphones

Android = Linux kernel + Dalvik VM (Google)

Routers y IoT

OpenWrt, DD-WRT, dispositivos embebidos

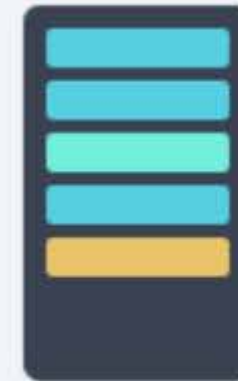
Satélites / Espacio

ISS, vehículos espaciales y drones militares

Linux: En todas partes



Supercomputer



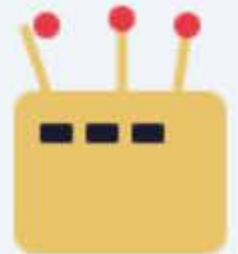
Servidor



Desktop



Smartphone



Router/IoT

Linux ejecuta el 96.4% de los
Top 500 supercomputadores

Servidores · Escritorio · Móviles · IoT · Satélites

96.4% Supercomputadores

72% Servidores Web

1.3 — Elegir un Sistema Operativo

Factores de Decisión

Función del equipo

Desktop, servidor, sistema embebido o contenedores.

Coste y licencias

Linux gratuito vs Windows/macOS con licencia comercial.

Ciclo de vida (LTS vs rolling)

Entorno empresarial = ciclo largo. Developer = ciclo corto.

Software requerido

¿Necesita aplicaciones específicas de Windows/Oracle/HP?

Habilidades del equipo

Formación disponible y curva de aprendizaje.

Seguridad y estabilidad

Linux: menor superficie de ataque, código auditado.

Windows	macOS	Linux
		
❑ Software masivo ❑ Fácil de usar ❑ Compatibilidad ❑ Licencia de pago ❑ Código cerrado	❑ UNIX certificado ❑ Diseño premium ❑ Creativo/Video ❑ Solo hardware Apple ❑ Alto coste	❑ Gratuito ❑ Código abierto ❑ Servidores/IoT ❑ Seguro y estable ❑ Curva aprendizaje
Server/Desktop	Principalmente Desktop	Server/Desktop/Móvil

Comparativa de Sistemas Operativos

1.3.1 — Ciclos de Liberación y Mantenimiento



Beta

Funciones nuevas no probadas en producción. Riesgo alto, máximas prestaciones.

Estable

Probada en campo. Recomendada para servidores y entornos críticos.

LTS (Long Term Support)

Ubuntu 24.04 LTS = soporte hasta 2029. Ideal para entornos empresariales.

EOL (End of Life)

Sin actualizaciones de seguridad. Migración obligatoria en producción.

1.3.7 — Android: La distribución Linux más popular

Arquitectura de Android

APLICACIONES (Apps de usuario)

ANDROID FRAMEWORK (APIs Java)

ART / DALVIK (Máquina Virtual)

LIBRERÍAS NATIVAS (C/C++)

KERNEL LINUX (modificado por Google)

Diferencias con GNU/Linux

Sin GNU Coreutils

Android no incluye las utilidades GNU (ls, cp, mv...) que usamos en servidor.

Sin Xorg / Wayland

Usa su propio sistema gráfico (SurfaceFlinger), incompatible con X11.

Dalvik / ART VM

Las apps corren en una máquina virtual Java-like, no directamente sobre el kernel.

BusyBox (solución)

Herramienta que proporciona versiones reducidas de los comandos UNIX en Android.

Google Play Store

Ecosistema cerrado de distribución de apps. Incompatible con repos APT/RPM.



Actividad de Repaso

1. ¿Qué diferencia hay entre el kernel Linux y una distribución Linux?

2. ¿Por qué se dice que Linux es «UNIX-like» pero NO es UNIX?

3. Explica la función del planificador de procesos (scheduler) del kernel.

4. ¿Qué ventajas tiene el modelo de código abierto frente al código cerrado?

5. Clasifica estos comandos según su tipo: `cd`, `ls`, `ll` (alias), `grep`, `mi_funcion()`

6. ¿En qué situación elegirías RHEL sobre Ubuntu? Justifica tu respuesta.

7. ¿En qué se diferencia Android de una distribución GNU/Linux de escritorio?

Resumen del Capítulo 1

🐧 Linux = kernel de código abierto (GPL) — Torvalds, 1991

⚙️ Kernel: gestiona CPU, RAM, I/O y procesos (planificador)

📦 Distribución = Kernel + GNU + apps + gestor de paquetes

● Familia Red Hat: RHEL, Fedora, CentOS — gestor RPM

● Familia Debian: Debian, Ubuntu, Mint — gestor APT/DEB

💻 Tipos de comando: builtin, externo, alias, función

🌐 Linux ejecuta desde supercomputadores hasta smartphones

📱 Android = Linux kernel + ART VM (sin GNU tools)



***"Talk is cheap.
Show me the code."***

— Linus Torvalds