



# Universidad Nacional Autónoma de México



Facultad de Ingeniería

## Sistemas Operativos

*UNIX Y SU PAPEL EN LA HISTORIA DE LOS SISTEMAS  
OPERATIVOS.*

Acosta Jacinto Alan, "320101179"

Rubio Carmona Jose Angel. "320118937"

Sexto semestre.

Fecha de entrega : 21 de abril de 2025

Profesora: Gunnar Eyal Wolf Iszaevich

*Semestre 2025-2*

Ciudad de México, 21 de abril de 2025.

## 1. Introducción – 1.5 minutos

- Presentación del equipo:

**“Hola, somos Alan y Ángel, y en este video vamos a hablarles sobre uno de los sistemas operativos más importantes e influyentes de la historia: UNIX.”**

- Motivación:

**“¿Por qué hablar de UNIX hoy en día? Porque gran parte de la tecnología moderna está construida sobre sus ideas: desde servidores hasta celulares.”**

**Hablar de Unix hoy en día es hablar del origen de muchas de las tecnologías que usamos a diario. Aunque nació a finales de los años 60, sus principios siguen presentes en sistemas modernos como Linux, macOS, Android, e incluso en servidores y supercomputadoras que sostienen internet. Su diseño simple, modular y portátil revolucionó la forma en que se construyen los sistemas operativos. Comprender Unix es clave para entender la base técnica de la informática actual y el desarrollo de software moderno.**

Una frase muy representativa sobre el legado de Unix es de **Dennis Ritchie**, uno de sus creadores:

**"Unix is simple. It just takes a genius to understand its simplicity."**  
— Dennis Ritchie

**Traducción:**

*"Unix es simple. Solo se necesita un genio para entender su simplicidad."*

Esta frase resume perfectamente la filosofía detrás de Unix: aunque su diseño es minimalista y modular, su impacto ha sido profundo y duradero.

## 2. Historia de UNIX – 5 minutos

- Contexto histórico (1960s-1970s):

**Como parte de este acuerdo, AT&T dirigió una pequeña fracción de sus ingresos a Bell Labs, con el propósito expreso de mejorar las comunicaciones. Monopolio , Transistor**

**CTSS ("Compatible Time-Sharing System")** fue el primer sistema en demostrar que una computadora podía servir a múltiples personas al mismo tiempo de manera interactiva y eficiente. Fue un pionero en el desarrollo de sistemas operativos modernos, y una pieza clave en la genealogía de Unix.

Cada tipo de computadora tenía su propio lenguaje ensamblador, por lo que los sistemas operativos eran programas de lenguaje ensamblador grandes y complicados, cada uno escrito en el lenguaje específico de su propio hardware.

Esta falta de uniformidad entre los sistemas y el uso de lenguajes de bajo nivel incompatibles dificultaron enormemente el progreso porque requería múltiples versiones de programas: un programa escrito para un sistema operativo tenía que ser reescrito desde cero para pasar a un sistema operativo o arquitectura diferente.

- **MULTICS y sus complicaciones.**

Bell Labs, que tenía una gran experiencia en la creación de sus propios sistemas desde principios de la década de 1950, iba a colaborar en el sistema operativo.

Multics era intrínsecamente una perspectiva desafiante, y pronto se encontró con problemas. En retrospectiva, fue en parte víctima del segundo efecto del sistema: después de un éxito (como CTSS). La frase "sobre-ingeniería" aparece en varias descripciones, y Sam Morgan la describió como "un intento de trepar a demasiados árboles a la vez".

Desde la perspectiva de los Laboratorios Bell, en 1968 estaba claro que, aunque Multics era un buen entorno informático para el puñado de personas a las que apoyaba, no iba a lograr su objetivo. En consecuencia, Bell Labs abandonó el proyecto en abril de 1969, dejando que MIT y GE siguieran adelante.

En 1969, el panorama de los sistemas operativos era muy diferente. Sistemas como Multics eran **complejos y costosos**, mientras que Unix fue diseñado para ser **simple, eficiente y flexible**. El nombre "Unix" proviene de UNICS, una versión reducida de Multics, ideada por Thompson y Ritchie.

- **Ken Thompson y Dennis Ritchie en Bell Labs.**

## 1. Ken Thompson

Ken Thompson comenzó a trabajar en Bell Labs en 1966, inicialmente colaborando en el proyecto Multics, un sistema operativo ambicioso que resultó ser demasiado complejo y

costoso. Después de que Bell Labs se retiró de Multics en 1969, Thompson decidió continuar explorando el diseño de sistemas operativos por su cuenta.

Su oportunidad surgió al encontrar un PDP-7 sin uso, una computadora modesta pero con una buena interfaz gráfica. En ella escribió un juego de exploración espacial, pero más importante aún, usó esta experiencia para comenzar a diseñar un sistema operativo más simple, eficiente y útil: Unix.

◆ Sus principales aportaciones:

- Diseñó el núcleo original de Unix en lenguaje ensamblador para el PDP-7.
- Propuso el modelo de herramientas pequeñas combinables, que se convirtió en la filosofía Unix.
- Fue pionero en implementar el sistema de archivos jerárquico y la noción de “todo es un archivo”.
- Más adelante, trabajó junto a Joe Condon en el desarrollo de Belle, la primera computadora en alcanzar el nivel de maestro en ajedrez.

## 2. Dennis Ritchie

Dennis Ritchie se unió formalmente a Bell Labs en 1967 como parte del Computing Science Research Center. Inicialmente trabajó también en Multics, pero su mayor contribución llegó poco después: el diseño del lenguaje de programación C.

◆ Sus principales aportaciones:

- Co-creador de Unix junto a Thompson, especialmente a partir de la migración del sistema al PDP-11.
- En 1973, reescribió Unix casi por completo en lenguaje C, lo que convirtió al sistema en portable: podía ser modificado para funcionar en distintas arquitecturas sin reescribirlo desde cero.
- Escribió el manual de referencia del lenguaje C y coautor del influyente libro *The C Programming Language* junto a Brian Kernighan.
- Ayudó a estandarizar C, que se convirtió en uno de los lenguajes más utilizados en programación de sistemas.

- Fue jefe del departamento de Sistemas de Software en Bell Labs, liderando el desarrollo del sistema operativo Plan 9.

## Reconocimientos conjuntos

Ken Thompson y Dennis Ritchie recibieron varios premios internacionales por su trabajo, incluyendo:

- **Premio Turing (1983).**
- **Medalla Nacional de Tecnología (1999).**
- **Premio Japón en Tecnología de la Información y Comunicaciones (2011).**
- **Ingreso póstumo de Ritchie al Salón Nacional de Inventores (2019)**
  - **Primer UNIX en un PDP-7.**

Varios investigadores quedaron con ideas sin concretar pero con una fuerte motivación por desarrollar algo más simple y eficiente. Uno de ellos fue Ken Thompson, quien había trabajado en el diseño de sistemas en Multics y quería explorar por su cuenta un nuevo enfoque de sistema operativo.

Thompson encontró un equipo PDP-7 (Programmed Data Processor 7), una computadora de 18 bits fabricada por Digital Equipment Corporation (DEC) que estaba prácticamente en desuso dentro de Bell Labs. Aunque limitada en potencia para los estándares de la época, el PDP-7 tenía una interfaz gráfica decente y suficientes capacidades para experimentar con software de sistema.

¿Por qué el PDP-7?

- Tenía solo 8 KB de memoria principal (unas 8 mil palabras de 18 bits).
- Era una máquina barata, ya disponible, que no requería aprobación presupuestaria para usar.
- Su hardware era lo suficientemente flexible como para probar ideas experimentales, como sistemas de archivos, multitarea básica y llamadas al sistema.

Ken Thompson, con el apoyo ocasional de Dennis Ritchie, comenzó a escribir desde cero un sistema operativo simple, con las ideas clave de lo que luego se convertiría en Unix:

- Sistema de archivos jerárquico.
- Separación de procesos.
- Herramientas pequeñas y reutilizables.
- Manejo de archivos como flujos de bytes (todo es un archivo).

Para aprovechar el PDP-7, Thompson incluso programó un juego de exploración espacial como primera prueba gráfica del sistema. Aunque fue una distracción personal, ese juego sirvió como excusa para comenzar el desarrollo del entorno Unix, ya que necesitaba herramientas de programación, sistema de archivos y multitarea para hacerlo funcionar.

---

### Desarrollo técnico

El primer Unix en el PDP-7 fue escrito completamente en lenguaje ensamblador, ya que no existía aún el lenguaje C. El sistema incluía:

- Un núcleo básico (kernel) capaz de manejar archivos y procesos.
- Un intérprete de comandos rudimentario (shell).
- Un sistema de archivos con estructura jerárquica (directorio raíz `/`, subdirectorios).
- Programas como, `ls`, `cat`, `rm`, y un editor de texto muy primitivo.

Este entorno era muy básico, pero funcional, y permitía por primera vez en Bell Labs que un sistema operativo respondiera a varios usuarios de forma interactiva, aunque uno a la vez (todavía no era multiusuario plenamente).

---

### Transición hacia el PDP-11

El éxito del experimento en el PDP-7 motivó al grupo a solicitar una computadora más potente. Aunque la dirección no estaba interesada en financiar otro gran sistema como Multics, aceptaron la propuesta de adquirir un PDP-11, más moderno y económico. Esta decisión fue clave para la expansión de Unix: en el PDP-11, Unix fue reescrito en lenguaje C (por Dennis Ritchie), lo que lo hizo portable y escalable a futuro.

---

## Frase destacada

*“Unix fue creado con cero presupuesto, un equipo olvidado, y la necesidad de jugar un videojuego.”*

— Parafraseando a Brian Kernighan

## Impacto del PDP-7 en Unix

- Permitió la primera implementación funcional de Unix.
- Demostró que un sistema operativo podía ser eficiente, modular y escrito por pocas personas.
- Sirvió como laboratorio de pruebas para ideas que se mantienen vigentes hasta hoy.
  - Reescritura en C: pionera en portabilidad.
- Importancia del lenguaje C:

## El cambio histórico: del ensamblador al C

En 1972, Dennis Ritchie comenzó a desarrollar un nuevo lenguaje de programación llamado C, basado en su experiencia previa con lenguajes como B y BCPL. El objetivo era crear un lenguaje de alto nivel, eficiente y lo suficientemente cercano al hardware como para escribir sistemas operativos, pero sin caer en la rigidez del ensamblador.

En 1973, junto a Ken Thompson, Ritchie logró una hazaña sin precedentes: reescribieron casi todo el sistema Unix en C (excepto partes críticas del núcleo relacionadas con el hardware, que quedaron en ensamblador). Este evento marcó un punto de inflexión en la historia del desarrollo de software.

---

## Importancia del lenguaje C

### 1. Portabilidad

La reescritura en C permitió que Unix pudiera ser portado fácilmente a otras arquitecturas sin necesidad de comenzar desde cero. Esto fue revolucionario: por primera vez, un sistema operativo podía adaptarse a diferentes máquinas con cambios mínimos.

Esto convirtió a Unix en uno de los primeros sistemas operativos portables de la historia.

### 2. Eficiencia y control

Aunque es un lenguaje de alto nivel, C permite acceder directamente a la memoria, manejar bits, bytes y registros, lo cual lo hace ideal para programación de bajo nivel, como la requerida por un sistema operativo.

### 3. Base de sistemas modernos

El lenguaje C no solo permitió el desarrollo de Unix, sino que se convirtió en el lenguaje de referencia para sistemas operativos y software de infraestructura. De hecho:

- Linux, macOS y Windows contienen grandes partes escritas en C.
- Lenguajes como C++, Objective-C y Rust derivan o se inspiran en su diseño.

### 4. Legado educativo y profesional

C se convirtió también en un estándar en la enseñanza de la programación a nivel universitario y sigue siendo imprescindible en áreas como sistemas embebidos, controladores de hardware, firmware y ciberseguridad.

---

Frase destacada de Dennis Ritchie:

*“C is quirky, flawed, and an enormous success.”*  
— Dennis Ritchie

---

### Conclusión

La decisión de reescribir Unix en C no solo mejoró su eficiencia, sino que le dio vida más allá de una sola máquina. Gracias a esa visión, Unix pudo evolucionar y propagarse rápidamente por el mundo académico y luego comercial, dando origen a los sistemas operativos modernos. El lenguaje C y Unix evolucionaron juntos, y ambos siguen siendo pilares fundamentales de la informática actual.

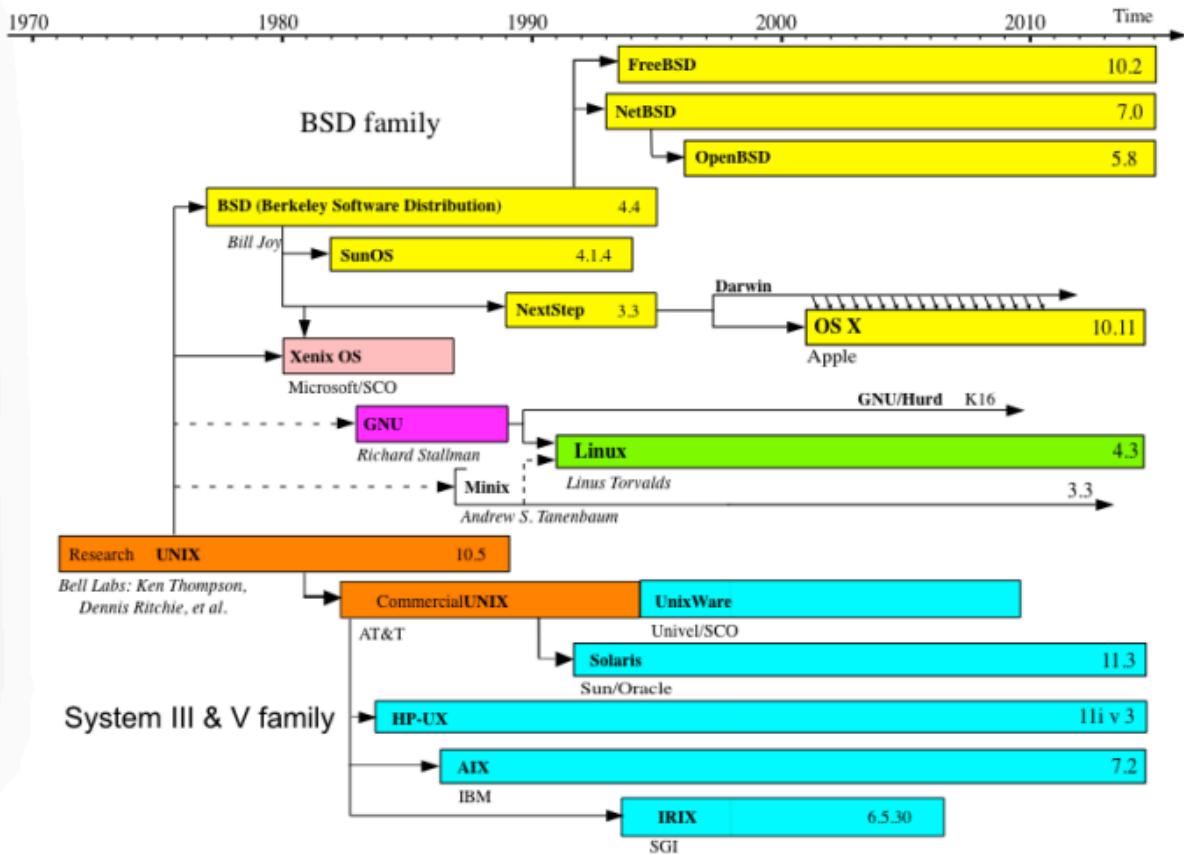
- **Expansión académica y comercial:**
  - **Distribución a universidades como Berkeley.**
  - **Nacimiento de BSD.**
  - **Aparición de variantes comerciales: System V, Solaris, etc.**

En la década de 1970, Bell Labs (AT&T) no podía comercializar Unix directamente debido a **restricciones legales impuestas por su condición de monopolio en telecomunicaciones**. Sin

embargo, esto permitió distribuir Unix libremente a instituciones educativas bajo licencias de muy bajo costo.

Uno de los principales beneficiarios fue la **Universidad de California en Berkeley (UC Berkeley)**, donde Unix llegó por primera vez en **1974**, en su versión **Unix Version 6 (V6)**. Gracias al acceso al código fuente, los estudiantes e investigadores comenzaron a **modificar, mejorar y extender el sistema**, contribuyendo enormemente a su evolución.

- Pequeño cuadro visual de la evolución de UNIX → Linux, macOS, Android.



#### 4. Sistema de Archivos – 3 minuto

Es la estructura que organiza los datos en una partición de disco. Se basa en un modelo arborescente y recursivo, en donde los nodos pueden ser tanto archivos como directorios, y estos últimos pueden contener a su vez directorios o subdirectorios.

En Unix todo se considera como un archivo. Esto permite que se pueda leer o escribir a un dispositivo.

- Estructura jerárquica ("")

## Estructura jerárquica – raíz (/)

- El sistema de archivos de Unix tiene una estructura jerárquica en forma de árbol, donde todo comienza desde un único punto llamado directorio raíz, representado por /.
- Desde la raíz, se ramifican todos los demás directorios y archivos. No hay letras de unidad (como en Windows con C:, D:...), sino una única estructura continua.

### 💡 Ejemplo visual de la jerarquía

```
css

/
├── bin      → Programas básicos del sistema
├── etc      → Archivos de configuración
├── home     → Directorios personales de los usuarios
│   └── alice
│       └── documentos.txt
├── usr      → Aplicaciones y bibliotecas instaladas
├── var      → Archivos variables (logs, colas de impresión)
└── tmp      → Archivos temporales
```

### 📌 Ejemplos clave de directorios

Directorio	Función
/	Raíz del sistema, punto de partida de todo.
/home	Contiene los archivos y carpetas personales de cada usuario. Ej: /home/alice
/etc	Archivos de configuración del sistema. Ej: configuración de red, servicios.
/usr	Programas de usuario y bibliotecas compartidas. Subdividido en /usr/bin, /usr/lib, etc.

### • Tipos de archivos:

Una de las ideas fundamentales de Unix es que “todo es un archivo”. Esto significa que, además de documentos de texto o programas, Unix trata directorios, dispositivos, y hasta conexiones de red como archivos, lo que unifica la forma de interactuar con el sistema.

ORDINARIOS – Contienen los datos de los usuarios (ASCII o binarios)

### ¿Qué son?

Son los archivos más comunes, que contienen **datos creados o usados por los usuarios y programas**.

### **Contenido:**

- **Texto plano (ASCII):** scripts, archivos de configuración, código fuente.
- **Binarios:** programas ejecutables, imágenes, archivos multimedia, datos en formatos no legibles directamente.

DIRECTORIOS – Contienen archivos.

### **¿Qué son?**

Son archivos especiales que **almacenan referencias a otros archivos y directorios**.

Son la base del **sistema de archivos jerárquico** de Unix.

### **Contenido:**

Listas de entradas que apuntan a otros archivos mediante **inodos** (estructuras internas que almacenan metadatos de archivos).

ESPECIALES Representan rutinas del kernel que proporcionan acceso a diversos dispositivos del sistema;

Estos **no almacenan datos comunes**, sino que representan **interfaces con el kernel** para interactuar con dispositivos o procesos. Se ubican comúnmente en el directorio `/dev`.

– Bloque.

### **¿Qué son?**

Dispositivos que transfieren datos en **bloques de tamaño fijo**.

### **Ejemplos:**

Discos duros, memorias USB, CD-ROMs.

`/dev/sda` (primer disco duro)

– Carácter.

### **¿Qué son?**

Dispositivos que transfieren datos en **flujos de caracteres**.

### **Ejemplos:**

Teclado, mouse, terminales, puertos serie.

- `/dev/tty`, `/dev/random`

– FIFO (pipes)

### ¿Qué son?

Canales especiales para comunicación entre procesos (**IPC**), donde un proceso escribe y otro lee.

### Uso común:

Comunicación entre programas sin archivos temporales.

### Ejemplo:

Un proceso escribe datos, otro los lee en orden.

– Sockets

### ¿Qué son?

Archivos especiales que permiten la **comunicación entre procesos a través de redes o dentro del sistema local (IPC)**.

### Usos:

Comunicación cliente-servidor, conexión de bases de datos, servidores web, etc.

### Ejemplo:

`/var/run/docker.sock` – usado para comunicarse con el daemon de Docker.

– Ligas simbólicas

### ¿Qué son?

Archivos especiales que **actúan como accesos directos** apuntando a otro archivo o directorio.

**Si el archivo original desaparece, el enlace se rompe.**

**Muy útiles para mantener compatibilidad entre rutas, versiones o configuraciones.**

Resumen

telnet ya no se usa por que no esta encriptado

ssh

## Tipos de archivos

- **SÍMBOLO SIGNIFICADO**

- Ordinario

- d Directorio

- c Carácter

- b Bloque

- l Liga simbólica

- s Socket

- p Pipe

Unix utiliza diferentes tipos de archivos no solo para **almacenar información**, sino para **interactuar con el sistema operativo, dispositivos físicos, otros procesos y servicios**. Esta unificación bajo el modelo de “todo es un archivo” permite una **interfaz simple y potente**, y ha sido una de las razones del **éxito y longevidad del sistema Unix y sus derivados** como Linux y macOS.

### **UNIX EN MAC**

#### **COMANDOS**

UNICS -Monousuario y monotarea

brian Kernighan

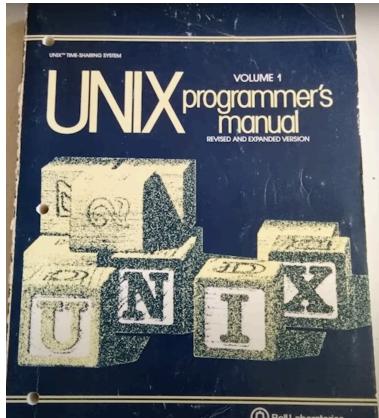
Luego pasó a UNIX

Los comandos básicos de unix, si se dan cuenta los comandos son muy similares a los que se utilizan en Linux. Esto se debe a que Linux tiene sus raíces en Unix, y muchos de sus comandos provienen directamente de la misma base. De hecho, una de las principales características de ambos sistemas operativos es su filosofía de herramientas pequeñas y especializadas que se pueden combinar para realizar tareas complejas. Esta compatibilidad entre comandos permite que los usuarios que dominan Unix puedan moverse fácilmente entre ambos sistemas.



Comando “man”

Este es un comando el cual sustituye la guia UNIX “programmer's manual”



Este se ocupa como man “comando deseado”.

```
rubio@MacBook-Air-de-Jose: ~ % man ls
```

```
NAME
ls - list directory contents

SYNOPSIS
ls [-@ABCDFGHILOPRSTUWabcdefghijklmnopqrstuvwxyz1%] [--color=when] [-D format] [file ...]

DESCRIPTION
For each operand that names a file of a type other than directory, ls displays its name as well as any requested, associated information. For each operand that names a file of type directory, ls displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

The following options are available:

-@      Display extended attribute keys and sizes in long (-l) output.
-A      Include directory entries whose names begin with a dot ('.') except for '.' and '..'. Automatically set for the super-user unless -I is specified.
-B      Force printing of non-printable characters (as defined by ctype(3) and current locale settings) in file names as \xxx, where xxx is the numeric value of the character in octal. This option is not defined in IEEE Std 1003.1-2008 ("POSIX.1").
-C      Force multi-column output; this is the default when output is to a terminal.
-D format
      When printing in the long (-l) format, use format to format the date and time output. The argument format is a string used by strftime(3). Depending on the choice of format string, this may result in a different number of columns in the output. This option overrides the -T option. This option is not defined in IEEE Std 1003.1-2008
```

Unix al inicio fue creado como un entorno sencillo y de manera modular(Es decir varios módulos que trabajan en conjunto conectados a través de PIPES )

Ventajas de ocupar UNIX antes de otro sistema convencional.

## PIPELINES

Un pipeline es una forma de conectar comandos entre sí, de modo que la salida de uno se convierte en la entrada del siguiente.

Se conectan con “|” (pipe).

El comando de la izquierda produce salida.

El comando de la derecha la consume como entrada.

Se pueden encadenar tantos como quieras.

```
rubio@MacBook-Air-de-Jose Expo % cat > servidor.log <<EOF
2025-04-14 08:00 INFO conexión establecida con 192.168.1.1
2025-04-14 08:05 ERROR conexión fallida con 192.168.1.2
2025-04-14 09:10 INFO conexión establecida con 192.168.1.3
2025-04-14 10:45 ERROR conexión fallida con 192.168.1.4
2025-04-14 11:00 ERROR conexión fallida con 192.168.1.5
2025-04-13 17:30 ERROR conexión fallida con 192.168.1.6
2025-04-14 13:00 INFO conexión establecida con 192.168.1.7
2025-04-14 13:15 ERROR conexión fallida con 192.168.1.8
2024-04-14 15:30 ERROR conexión fallida con 192.168.1.9
EOF
rubio@MacBook-Air-de-Jose Expo % cat servidor.log | grep "ERF
4
rubio@MacBook-Air-de-Jose Expo %
```

### **Comando GREP**

Es una herramienta para **buscar texto que coincida con un patrón o palabra**, ya sea en uno o más archivos.

Buscar palabras clave en archivos de texto

- Filtrar resultados en logs
- Encontrar errores, nombres, valores, etc.
- Usarlo con otras herramientas (pipes)

```
rubio@MacBook-Air-de-Jose Texto.txt % touch Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % echo "Aqui se encuentra dos veces la palabra error (error)" > Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % cat Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % grep 'error' Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % echo "Aqui no se encuentra la palabra" >> Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep 'error' Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % grep 'palabra' Error.txt
Aqui se encuentra dos veces la palabra error (error)
Aqui no se encuentra la palabra
rubio@MacBook-Air-de-Jose Texto.txt % echo "Hola" >> Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep 'Hola' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'o' Error.txt
Aqui se encuentra dos veces la palabra error (error)
Aqui no se encuentra la palabra
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'H' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'h' Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep -i 'h' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt %
```

### **Comando Sed**

Stream EDitor (editor de flujo)

Sirve para modificar texto automáticamente, línea por línea, sin abrir el archivo manualmente.

- Reemplazar palabras o frases
- Eliminar líneas específicas
- Insertar o modificar contenido
- Automatizar limpieza de datos

```
rubio@MacBook-Air-de-Jose Texto.txt % echo probando el comando sed  
probando el comando sed  
[rubio@MacBook-Air-de-Jose Texto.txt % sed 's/#/aqui/' Error.txt  
  
Solo remplaza la primer ocurrencia (aqui, # ,#)  
  
Remplaza todas las ocurrencias(%, %, %)  
  
[Las remplaza y lo guarda (cambio, cambio )%  
rubio@MacBook-Air-de-Jose Texto.txt % sed 's/%/aqui/g' Error.txt  
  
Solo remplaza la primer ocurrencia (#, # ,#)  
  
Remplaza todas las ocurrencias(aqui, aqui, aqui)  
  
[Las remplaza y lo guarda (cambio, cambio )%  
rubio@MacBook-Air-de-Jose Texto.txt % sed -i '' 's/cambio/Hola/g' Error.txt  
[rubio@MacBook-Air-de-Jose Texto.txt % cat Error.txt  
  
Solo remplaza la primer ocurrencia (#, # ,#)  
  
Remplaza todas las ocurrencias(%, %, %)  
  
[Las remplaza y lo guarda (Hola, Hola )%  
rubio@MacBook-Air-de-Jose Texto.txt %
```

## ***UNIX y su legado***

### **Linux (1991)**

Creado por **Linus Torvalds**, Linux es un sistema operativo inspirado en UNIX y basado originalmente en MINIX. Aunque **no está basado directamente en el código fuente de UNIX**, adopta su estructura, comandos, y filosofía. Hoy en día es el corazón de servidores, supercomputadoras, sistemas Android, dispositivos IoT y más.

- Fue creado como un proyecto libre y abierto.
- Representa el espíritu comunitario y descentralizado de UNIX.
- Es el **sistema operativo de código abierto más usado del mundo**.

### **Windows NT(1993)**

Desarrollado por Microsoft como una **respuesta empresarial a UNIX**, Windows NT fue diseñado desde cero, pero con muchas ideas inspiradas en los sistemas UNIX, como la multitarea, seguridad por usuarios y arquitectura modular.

- Su objetivo era **competir con UNIX en el entorno corporativo**.
- Aunque no es UNIX ni está basado en él, fue claramente influenciado por su éxito.
- Dio origen a todas las versiones modernas de Windows (2000, XP, 7, 10...).

### **Mac OS X (1997 en adelante)**

Después del regreso de Steve Jobs a Apple, el sistema operativo clásico de Mac fue reemplazado por **macOS**, basado en **UNIX BSD (Berkeley Software Distribution)** a través de NeXTSTEP.

- macOS es **UNIX certificado**.
- Es considerado uno de los **sistemas más estables y seguros**.
- Combina lo mejor de UNIX con una interfaz amigable.

### C y C++ (Origen y presente)

El lenguaje **C** fue creado por **Dennis Ritchie** para reescribir UNIX, lo que permitió su portabilidad a distintas máquinas.

- C es el **lenguaje madre de los sistemas operativos** modernos.
- C++ extendió C con programación orientada a objetos y es clave en desarrollo de software, videojuegos, motores gráficos y más.

Hoy en día, tanto C como C++ siguen siendo **lenguajes fundamentales** para programadores de bajo nivel, sistemas embebidos y aplicaciones de alto rendimiento.

**(el mas importante para el desarrollo del sistema)50 años despues los sistemas operativos y el hardware entre muchas se escriben con C y C++**

**AQUÍ**



# Guión extendido para presentación/video de 20 minutos

**Tema: UNIX: Historia, Arquitectura y Legado**

**Duración total estimada: 20 minutos**

---

## 1. Introducción – 1.5 minutos

- **Presentación del equipo:**

“Hola, somos Alan y Ángel, y en este video vamos a hablarles sobre uno de los sistemas operativos más importantes e influyentes de la historia: UNIX.”

- **Motivación:**

“¿Por qué hablar de UNIX hoy en día? Porque gran parte de la tecnología moderna está construida sobre sus ideas: desde servidores hasta celulares.”

---

## 2. Historia de UNIX – 5 minutos

- **Contexto histórico (1960s-1970s):**

- MULTICS y sus complicaciones.
- Ken Thompson y Dennis Ritchie en Bell Labs.
- Primer UNIX en un PDP-7.
- Reescritura en C: pionera en portabilidad.

- **Importancia del lenguaje C:**

“Antes, los sistemas operativos eran escritos en lenguaje ensamblador. UNIX fue de los primeros en cambiar eso, usando un lenguaje más accesible y transportable: el C.”

- **Expansión académica y comercial:**

- Distribución a universidades como Berkeley.
- Nacimiento de BSD.
- Aparición de variantes comerciales: System V, Solaris, etc.

- Pequeño cuadro visual de la evolución de UNIX → Linux, macOS, Android.
- 

### 3. Arquitectura de UNIX – 4 minutos

- **Modelo en capas:**

- Kernel → Shell → Programas de usuario.
- Breve analogía: Kernel es el motor del coche, Shell es el volante.

- **El Kernel:**

- Maneja memoria, procesos, dispositivos, archivos.
- Concepto de espacio de usuario vs espacio de kernel.
- Se encarga de gestionar los recursos de hardware, como la memoria, el procesador, el sistema de archivos y los dispositivos.

- **El Shell:**

- Es la interfaz entre el usuario y el kernel. Interpreta los comandos introducidos por el usuario y los traduce en llamadas al sistema.
- CLI (Command Line Interface).
- Shells famosos: Bourne Shell, Bash, Zsh.

- **Ejemplo completo paso a paso:**

- Usuario escribe `rm archivo.txt`.

- Shell interpreta.
  - Kernel verifica permisos y ejecuta la eliminación.
- 

## 4. Sistema de Archivos – 3 minutos

- **Estructura jerárquica ("")**
    - Ejemplo visual de directorios: `/`, `/home`, `/etc`, `/usr`.
  - **Tipos de archivos:**
    - Archivos normales
    - Directorios
    - Enlaces simbólicos y duros
    - Dispositivos como archivos (`/dev/sda`, etc.)
  - **Permisos y propietarios:**
    - `rwx` (read, write, execute)
    - Usuario, grupo, otros.
    - Ejemplo: `chmod +x script.sh`
- 

## 5. Comandos Básicos – 4.5 minutos

- Comandos más comunes y útiles:
  - `ls`, `cd`, `pwd`, `cp`, `mv`, `rm`, `mkdir`, `rmdir`.
  - `grep`, `cat`, `less`, `tail`, `head`.
- **Uso de pipes y redirección:**
  - Explicación de `|`, `>`, `>>`, `<`.

## Ejemplo:

bash

CopiarEditar

```
cat log.txt | grep "ERROR" > errores.txt
```

○

- **Scripts en shell:**

- Mostrar un pequeño script en Bash.
- Explicar variables, condicionales, loops.

- **Interactividad:**

- Mostrar cómo hacer un script que pida al usuario su nombre y lo salude.
- 

## 6. Legado y actualidad – 1.5 minutos

- Sistemas modernos basados en UNIX:

- Linux, macOS, BSD, Android (Linux Kernel).

- UNIX en servidores, supercomputadoras y móviles.

- Filosofía UNIX:

- “Haz una cosa y hazla bien.”
  - Uso de herramientas pequeñas encadenadas (composición).
- 

## 7. Conclusión – 1 minuto

- Resumen de ideas clave:

- UNIX como base de muchos sistemas modernos.
- Su arquitectura clara y modular.

- Su impacto en la filosofía de desarrollo.
- Cierre del video:

“Gracias por ver nuestro video. Esperamos que hayan aprendido más sobre UNIX y cómo su legado sigue vivo en la tecnología que usamos todos los días.”

## FUENTES

Kernighan, B. W. (2020). *UNIX: A history and a memoir*. Kindle Direct Publishing.

Hostinger Academy. (2021, 24 de febrero). *What is UNIX? UNIX Operating System Explained* [Video]. YouTube. <https://www.youtube.com/watch?v=KuGCq7L6WaM>

Hostinger. (n.d.). *50 Most Commonly Used Linux Commands (With Examples)*. Hostinger Tutorials. <https://www.hostinger.com/tutorials/linux-commands>

UNIX / Linux tutorial for beginners. (n.d.). Retrieved April 17, 2025, from Surrey.ac.uk website: <https://info-ee.surrey.ac.uk/Teaching/Unix/>

oLourdes Yolanda Flores Salgado, M. (n.d.). Introducción a UNIX. Retrieved April 17, 2025, from Unam.mx website: [https://triton.astroscu.unam.mx/fruiz/introunix\\_plan\\_2013.pdf](https://triton.astroscu.unam.mx/fruiz/introunix_plan_2013.pdf)

AprendoLibre. (2022, 1 de noviembre). *¿POR QUÉ NO APRENDO?* [Video]. YouTube. <https://www.youtube.com/watch?v=YvcPJlsfL8U>