




# COMPONENTES DE UN SISTEMA OPERATIVO GNU/LINUX

## MÓDULO II

Sistemas Abiertos – Administración de SO I



### Contenido del Módulo II.

- **Reseña histórica del Proyecto GNU.**
    - **Objetivos del Proyecto GNU.** ¿Quién es Richard Stallman? Creación de la FSF.
    - **Principales componentes.** Dificultades y obstáculos.
  - **El origen y el desarrollo de Linux.**
    - **Motivaciones y puntos de partida de Linus Torvals.** Desarrollo técnico.
    - **Publicación y distribución.** Diferencia entre kernel y SO completo.
  - **Composición de un SO GNU/Linux.**
    - **Adopción del kernel Linux por el Proyecto GNU.**
    - **Estructura general de GNU/Linux.**
    - **Estructura del kernel Linux.**
- 

# Reseña histórica del Proyecto GNU.

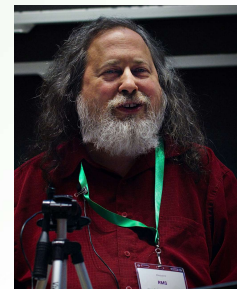
## • Introducción.

- Iniciado en 1983 por **Richard Stallman**, representa un hito fundamental en la historia del software libre.
- Nace con el objetivo de crear un SO completamente libre, compatible con Unix, pero sin las restricciones de licencias propietarias.
- GNU es un acrónimo recursivo: **GNU's not Unix.**
  - Busca ser compatible con Unix.
  - Pero sin ser Unix.
- Sentó las bases de lo que hoy se conoce como el Movimiento del Software Libre.

# Reseña histórica del Proyecto GNU.

## • ¿Quién es Richard Stallman?

- Nacido en 1953 en Nueva York, mostró desde una edad temprana una aptitud notable para la programación y la matemática.
- Su primera experiencia fue en el IBM Research Center, mientras cursaba la secundaria.
- Cuando se graduó, escribió otro programa, un preprocesador para el lenguaje de programación PL/1 en el IBM S/360.
- Se graduó en física en la Universidad de Harvard en 1974.
- Como programador, trabajó en el MIT en el laboratorio de IA.
- Es el principal de desarrollador del Proyecto GNU.
- Fundó la FSF el Proyecto GNU, inventó el concepto de copyleft.



## Reseña histórica del Proyecto GNU.

### • Objetivos del Proyecto GNU.

- El principal objetivo que tuvo el proyecto GNU fue el desarrollo de un SO completamente libre.
- Se basa en principios éticos y filosóficos, plasmados en las cuatro libertades:.
  - Uso.
  - Estudio y modificación.
  - Distribución.
  - Publicación de las mejoras.
- Las acciones o el trabajo del Proyecto GNU son los que propicia el MSF:
  - Desarrollar software libre.
  - Crear conciencia.
  - Legislación.

## Reseña histórica del Proyecto GNU.

### • Creación de la Free Software Foundation (FSF).

- Se funda en 1985 para apoyar el desarrollo y la promoción del software libre.
- Es una organización sin fines de lucro.
- Sirve como entidad legal para el proyecto GNU y para promover la adopción de software libre en la industria y en la sociedad en general.
- Es responsable de la creación y difusión de la Licencia Pública General de GNU (GPL).
- Además, proporciona recursos para el desarrollo de software libre:
  - Financiamiento de proyectos: iniciativa **Free Software Supporter**.
  - Infraestructura técnica: hosting y repositorios **Savannah**.
  - Promoción y difusión: evento **LibrePlanet**.
  - Educación y documentación: documentación y manuales técnicos.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **GCC (GNU Compiler Collection):**

- Uno de los logros más notables del proyecto GNU.
    - Originalmente desarrollado como el compilador de C para GNU.
    - Conjunto de compiladores para diversos lenguajes de programación, incluyendo C, C++, Objective-C, Fortran, Ada, y más.
    - En un estándar de facto en la programación y desarrollo de software libre.

- **Emacs:**

- Editor de texto altamente extensible y personalizable.
    - Se puede utilizar para:
      - Editar código fuente.
      - Gestionar proyecto.
      - Cliente de correo electrónico.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **glibc (GNU C Library):**

- Es la biblioteca estándar de C en los sistemas GNU.
    - Proporciona las interfaces básicas de programación para el núcleo del sistema operativo:
      - Manejo de archivos.
      - Administración de memoria.
      - Gestión de procesos.
    - Es fundamental para la compatibilidad del software, muchas aplicaciones dependen de ella para interactuar con el SO.
    - En un estándar de facto en la programación y desarrollo de software libre.
    - Su desarrollo ha sido vital para asegurar la portabilidad y funcionalidad.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **GDB (GNU Debugger):**

- Permite ver lo que ocurre dentro de un programa mientras se ejecuta o analizar lo que ha causado su fallo.
    - Soporta una amplia gama de lenguajes de programación.
    - Permite a los usuarios:
      - Detener la ejecución de programas.
      - Inspeccionar y modificar variables.
      - Seguimiento detallado del flujo de control del programa.
    - Es ampliamente utilizado en el desarrollo de software libre.
    - Ha sido crucial para la depuración y mejora de muchos proyectos importantes.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **Make:**

- Herramienta que automatiza la compilación y construcción de programas.
    - Lee archivos de configuración, conocidos como "Makefiles", que especifican cómo deben compilarse los diferentes módulos de un programa y en qué orden.
    - Es esencial en proyectos grandes, donde la gestión manual de la compilación sería impracticable.
    - Ha sido adoptado también en muchos proyectos comerciales, por su capacidad para manejar dependencias complejas y optimizar los procesos de construcción.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **Bash (Bourne Again Shell):**

- Es una de las contribuciones más conocidas del proyecto GNU al mundo de los sistemas operativos Unix y Unix-like.
    - Es un intérprete de comandos que proporciona una interfaz de línea de comandos.
    - Además de ser compatible con el shell Bourne original (sh), Bash incluye muchas características adicionales:
      - Historial de comandos.
      - Edición de líneas.
      - Completado de tabulaciones.
      - Capacidad de script avanzada.
    - Es el shell predeterminado en la mayoría de las distribuciones de GNU/Linux.

## Reseña histórica del Proyecto GNU.

- **Principales componentes desarrollados.**

- **Coreutils:**

- Conjunto de herramientas básicas que son esenciales para la manipulación de archivos, texto y procesos.
    - Incluye comandos como:
      - ls,
      - cp,
      - mv,
      - rm,
      - cat,
      - echo,
    - Estos forman el núcleo de la interacción con el SO a través de la línea de comandos.
    - Es el reemplazo de la versión propietaria de estos comandos.



## Reseña histórica del Proyecto GNU.

### • Principales componentes desarrollados.

#### • **Tar (tape archive):**

- Herramienta fundamental para la manipulación de archivos tarball.
- Se utiliza para combinar varios archivos en un solo archivo, útil para la distribución y archivado de software.
- No comprime, para ello se utiliza gzip o bzip2.
- No fue desarrollado por GNU originalmente, incluye mejoras que lo hacen más poderoso y flexible.

#### • **grep:**

- Herramienta utilizada para buscar texto dentro de archivos.
- Utiliza expresiones regulares.
- Permite buscar y filtrar grandes volúmenes de datos rápidamente
- Indispensable en el análisis de archivos de texto y en la gestión de logs.

## Reseña histórica del Proyecto GNU.

### • Dificultades y Obstáculos: El Caso del Kernel GNU Hurd.

- Una de las mayores dificultades que enfrentó el proyecto GNU fue la creación de un kernel completamente funcional, denominado GNU Hurd.
- La intención era crear un kernel basado en una arquitectura moderna y flexible, conocida como microkernel.
- Buscaba ofrecer mayor modularidad y estabilidad al sistema operativo.
- **El Enfoque del Microkernel (repaso de SO I):**
- GNU Hurd fue diseñado para funcionar sobre el microkernel Mach (MIT).
- La idea es mantener un núcleo pequeño con funciones en espacio de usuario.
- Esto proporciona mayor estabilidad y seguridad: una falla en un proceso en espacio de usuario no debería comprometer todo el sistema.

## Reseña histórica del Proyecto GNU.

### • Dificultades y Obstáculos: El Caso del Kernel GNU Hurd.

- **Dificultades Técnicas en el Desarrollo:**
- A pesar de las ventajas del microkernel, GNU Hurd enfrentó dificultades técnicas que dificultaron su desarrollo:
  - **Complejidad de la Comunicación Inter-Procesos (IPC):** eran más lentas y complicadas en comparación con las llamadas directas del sistema que se encuentran en los kernels monolíticos.
  - **Problemas de Sincronización y Concurrencia:** complicado de implementar, llevó a problemas de estabilidad.
  - **Falta de Recursos y Colaboradores:** a diferencia del kernel Linux, Hurd no logró generar un nivel similar de apoyo. El desafío era muy grande.
  - **Competencia con Linux:** Cuando Linux fue liberado en 1991, rápidamente se convirtió en una alternativa práctica y funcional. Esto desvió la atención y los recursos que podrían haberse dedicado a Hurd.

## Reseña histórica del Proyecto GNU.

### • Dificultades y Obstáculos: El Caso del Kernel GNU Hurd.

- **El Estado Actual de GNU Hurd:**
- El desarrollo de GNU Hurd no se ha detenido por completo.
- Un pequeño grupo de desarrolladores continúa trabajando en el proyecto.
- Ha alcanzado un nivel de funcionalidad que permite su uso en entornos experimentales.
- Sin embargo, sigue sin ser una opción viable debido a su rendimiento limitado y a la falta de soporte para muchas características modernas de hardware.
- La historia del desarrollo de Hurd ilustra los desafíos en la creación de un SO a partir de principios innovadores, pero también demuestra cómo la comunidad del software libre ha sido capaz de adaptarse y encontrar soluciones alternativas.



## El origen y desarrollo de Linux.

### • Motivaciones y Punto de Partida de Linus Torvalds.

- En 1991, Linus Torvalds, estudiante de 21 años de la Universidad de Helsinki, quería crear un SO para utilizar las características avanzadas de su computadora, un 386 de Intel, que en ese momento no soportaba Unix.
- Torvalds se inspiró en MINIX, un SO basado en Unix, creado por A. S. Tanenbaum para propósitos educativos.
- Aunque MINIX era útil para aprender sobre la estructura de un SO, Torvalds quería algo más robusto, que pudiera evolucionar sin las restricciones de MINIX.
- La motivación principal de Torvalds no era crear un SO para la comunidad, sino aprender y mejorar sus habilidades en programación y el funcionamiento de SO.
- Sin embargo, se dio cuenta que su proyecto podía ser de interés para otros y comenzó a compartir su código con la comunidad.

## El origen y desarrollo de Linux.

### • Desarrollo Técnico.

- Torvalds eligió C como el lenguaje para su proyecto, ya que era el estándar para el desarrollo de SO en ese momento.
- Además, C ofrecía la combinación ideal entre control de bajo nivel y portabilidad, lo que permitía un código eficiente y adaptable a diferentes arquitecturas de hw.
- Uno de los aspectos técnicos clave en el desarrollo de Linux fue la elección de una arquitectura monolítica para el kernel.
- A diferencia de los microkernels, un kernel monolítico incluye todas las funciones básicas del SO dentro de un único espacio de memoria.
- Esta elección fue pragmática; los microkernels ofrecían más estabilidad y modularidad, pero la monolítica era más simple y ofrecía un mejor rendimiento.
- El primer Linux, la versión 0.01, se hizo público en septiembre de 1991. Contenía unas 10.000 líneas de código y requería MINIX para la compilación.

## El origen y desarrollo de Linux.

### • **Publicación y Distribución.**

- El verdadero punto de inflexión en la historia de Linux fue cuando Torvalds decidió publicar el código bajo la Licencia GPL en 1992.
- Esta decisión alineó el proyecto Linux con la comunidad del software libre.
- Linux rápidamente atrajo a una comunidad global de desarrolladores que comenzaron a colaborar, mejorar y expandir el sistema.
- Torvalds gestionaba el proyecto a través de listas de correo, donde se discutía, se proponían cambios y enviaban parches para mejorar el código.
- Esto permitió que Linux evolucionara rápidamente, incorporando nuevas características, mejorando el soporte de hw, y corrigiendo errores a un ritmo mucho más rápido que cualquier sistema operativo propietario.

## El origen y desarrollo de Linux.

### • **Diferenciación entre kernel y sistema operativo completo.**

- Linux es, en esencia, un **kernel** y no un sistema operativo completo.
- Un kernel es la parte central de un SO, gestiona el hardware, la memoria, los procesos y la comunicación entre ellos.
- Por sí solo **no proporciona una GUI ni herramientas** que hacen lo funcional.
- Aquí es donde el proyecto GNU entra en escena: Linux se combina con las herramientas del proyecto GNU para formar **GNU/Linux**.
- Este término refleja la unión de dos esfuerzos: el kernel Linux y el conjunto de herramientas de GNU.
- Hoy en día, las distribuciones de GNU/Linux incluyen además **entornos de escritorio**, aplicaciones de usuario, servidores, etc.

## Composición de un SO GNU/Linux.

### • Adopción del Kernel Linux por el Proyecto GNU.

- En 1992, se abrió la puerta para combinar el kernel Linux con los componentes de GNU, creando así un SO completo y funcional conocido como GNU/Linux.
- La adopción de Linux como kernel del SO fue un paso crucial para el éxito del proyecto GNU.
- Resolvió la limitación más grande que tenían: la falta de un **kernel estable**.
- A partir de este punto, GNU/Linux se convirtió en la base de muchas distribuciones, que no solo eran completamente funcionales sino también libres y abiertas.

## Composición de un SO GNU/Linux.

### • Estructura general de GNU/Linux.

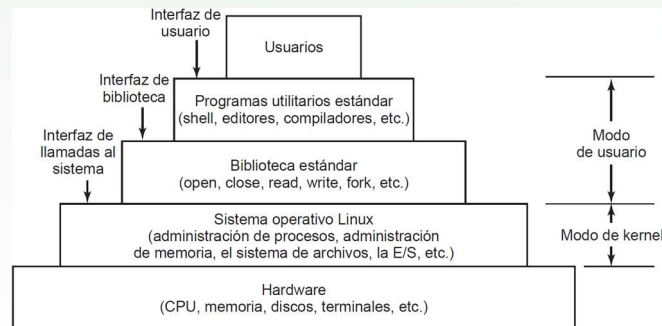
- Para comprender mejor la estructura de un SO GNU/Linux, podemos pensar en un modelo simplificado de capas.
- En este modelo, Linux se sitúa justo por encima del hw, en lo que llamamos el espacio del kernel, mientras que las herramientas de GNU se encuentran en el espacio de usuario, por encima de Linux.



# Composición de un SO GNU/Linux.

- **Estructura general de GNU/Linux.**

- Si afinamos este modelo, podemos visualizarlo como una pirámide.
- En la base, está el hw, que incluye CPU, memoria, discos, y dispositivos de E/S.
- Por encima del hw, se ejecuta el SO.



# Composición de un SO GNU/Linux.

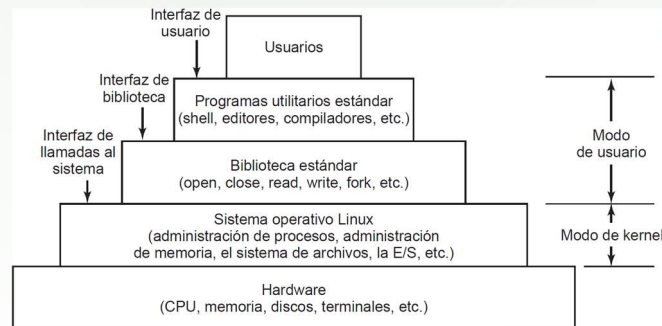
- **Estructura general de GNU/Linux.**

- Además del SO y la biblioteca de llamadas al sistema, todas las versiones de Linux incluyen una serie de programas estándar.
- Algunos de estos programas están definidos por el estándar **POSIX 1003.2**, mientras que otros varían según la distribución.
- Entre estos programas encontramos el **intérprete de comandos** (o shell), los **compiladores**, **editores de texto**, **programas de procesamiento de texto**, y **herramientas de gestión de archivos**.
- Estos son los programas que el usuario invoca directamente mediante el teclado.

# Composición de un SO GNU/Linux.

## • Estructura general de GNU/Linux.

- Por lo tanto, podemos identificar tres interfaces principales en GNU/Linux:
  - La interfaz de llamadas al sistema.
  - La interfaz de la biblioteca de funciones.
  - La interfaz de los programas utilitarios estándar, que el usuario ejecuta directamente.



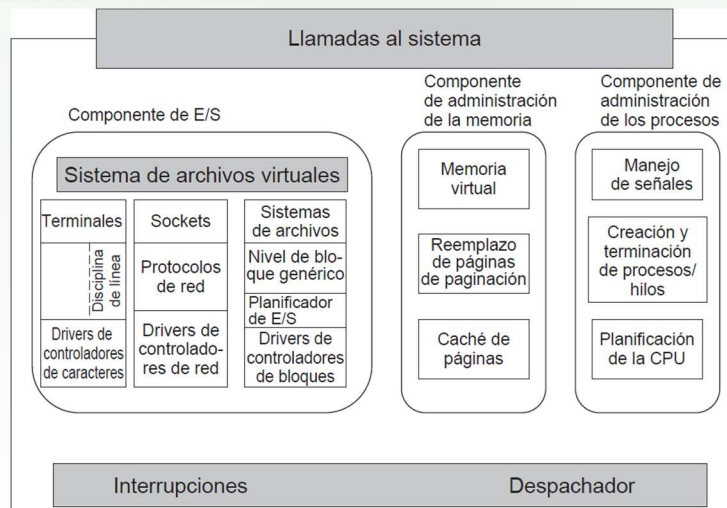
# Composición de un SO GNU/Linux.

## • Estructura del kernel Linux.

- El kernel de Linux se encuentra directamente sobre el hw y permite la interacción con la E/S, la memoria y la CPU.
- En su nivel más bajo, contiene manejadores de IRQ, que son la forma principal de comunicarse con los dispositivos, y el despachador, que es quien hace el cambio físico de procesos.
- No confundir con el planificador, ya que éste toma la decisión de qué proceso va a ejecutar.
- Cuando ocurre una interrupción, el kernel guarda el estado del proceso en ejecución e invoca el controlador apropiado para manejar el evento.
- Posteriormente, cuando el kernel completa ciertas operaciones, se reanuda la ejecución del proceso de usuario.

# Composición de un SO GNU/Linux.

- Estructura del kernel Linux.



## Fin del Módulo II