




# **SISTEMAS ABIERTOS y SOFTWARE LIBRE**

## **MÓDULO I**

Sistemas Abiertos – Administración de SO I



### **Contenido del Módulo I.**

- **Sistemas Abiertos.**
    - Definición. Características principales. Ejemplos.
    - Importancia. Impacto. Casos de éxito.
  - **Software Libre.**
    - Definición.
    - Las cuatro libertades.
  - **Historia y Filosofía del Movimiento del Software Libre.**
    - Origen y evolución.
    - Principios y valores promovidos por el movimiento.
    - El papel de la Free Software Foundation (FSF).
  - **Licencias de Software Libre.**
    - Diferencias entre licencias permisivas y copyleft.
    - Tipos de licencias: GPL, LGPL, MIT, Apache.
  - **Comparación con Software Propietario.**
    - Diferencias entre Software Libre y Software Propietario.
    - Ventajas y Desventajas de Cada Enfoque.
    - Impacto en la Industria y en los Usuarios.
- 

# Sistemas Abiertos.

## • Definición.

- Sistema informático que está diseñado para ser **compatible** e **interoperable** con otros sistemas.
- Utilizan **estándares abiertos** y **especificaciones públicas**.
- Promueven la **colaboración** y la **innovación**.
  - Diferentes tecnologías trabajan juntas de manera eficiente.
- La filosofía de los sistemas abiertos:
  - Transparencia.
  - Accesibilidad.
  - Flexibilidad.

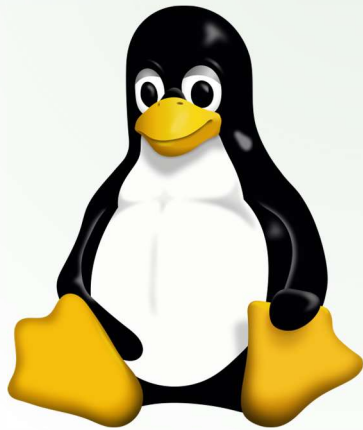
# Sistemas Abiertos.

## • Características principales.

- **Interoperabilidad:**
  - Funcionan con otros sistemas y tecnologías sin problemas.
- **Accesibilidad:**
  - Especificación completa del sistema disponible públicamente.
- **Flexibilidad:**
  - Libertad de modificar y personalizar el sistema.
- **Transparencia:**
  - Código fuente y especificaciones accesibles para cualquiera.
  - Auditorías independientes.
- **Evolución y mejora continua:**
  - Mejoras, correcciones y nuevas funcionalidades desarrolladas por colaboradores

## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Sistemas operativos.



## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Redes y protocolos de Internet.



## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Lenguajes de programación.



## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Sistemas Gestores de Bases de Datos.



## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Aplicaciones web y servidores.



NGINX



## Sistemas Abiertos.

- Ejemplos de SA en diferentes contextos tecnológicos.
  - Software de ofimática.



## Sistemas Abiertos.

### • Importancia de los Sistemas Abiertos.

- **Promoción de la innovación:** cualquier persona puede contribuir con nuevas ideas y mejoras.
- **Reducción de costos:** evitar los costos asociados con las licencias de software propietario.
- **Fomento de la competencia:** múltiples proveedores compitan en igualdad de condiciones.
- **Independencia y control:** no están atados a un solo proveedor para el soporte y las actualizaciones.
- **Seguridad y confiabilidad:** la transparencia permite auditorías independientes y revisiones por parte de la comunidad.
- **Colaboración global:** permiten la colaboración de desarrolladores y usuarios de todo el mundo.

## Sistemas Abiertos.

### • Impacto en la interoperabilidad y la innovación.

- **Interoperabilidad:**
  - **Estándares abiertos:** Facilita la comunicación y la integración entre diferentes sistemas y tecnologías. Esencial donde se requiere colaboración entre múltiples plataformas y dispositivos.
  - **Compatibilidad:** Garantizan que diferentes componentes puedan trabajar juntos sin problemas, reduciendo la fragmentación tecnológica y mejorando la eficiencia operativa.
- **Innovación:**
  - **Desarrollo colaborativo:** La colaboración global acelera el desarrollo de nuevas tecnologías y soluciones innovadoras.
  - **Ecosistema vibrante:** La comunidad activa crea un ecosistema dinámico donde las nuevas ideas pueden ser probadas y adoptadas rápidamente.
  - **Adopción de nuevas tecnologías:** Pueden adaptarse y evolucionar rápidamente para incorporar nuevas tecnologías y metodologías.



## Sistemas Abiertos.

- **Ejemplos de casos de éxito y adopción en la industria.**

- **Linux en servidores:**

- **Caso de éxito:** Es el SO más utilizado en servidores, debido a su robustez, seguridad y capacidad de personalización.
    - **Impacto:** Ha permitido construir infraestructuras escalables y fiables, impulsando su capacidad de innovación y su competitividad en el mercado.

- **Apache HTTP Server:**

- **Caso de éxito:** Es uno de los servidores web más utilizados, alojando millones de sitios web. Su éxito se debe a su estabilidad, seguridad y flexibilidad.
    - **Impacto:** Ha permitido ofrecer servicios web de alta calidad, contribuyendo significativamente al crecimiento de Internet.

- **Mozilla Firefox:**

- **Caso de éxito:** Es un navegador web de código abierto, adoptado por millones de usuarios. Su desarrollo colaborativo ha resultado en un navegador rápido, seguro e innovador.
    - **Impacto:** Ha influido en la industria de los web browsers al introducir nuevas tecnologías y estándares, fomentando una competencia saludable e impulsando la innovación.

## Sistemas Abiertos.

- **Ejemplos de casos de éxito y adopción en la industria.**

- **Android:**

- **Caso de éxito:** Basado en el núcleo de Linux, es el SO móvil más utilizado en el mundo. Ha permitido a fabricantes de dispositivos y desarrolladores de aplicaciones crear una amplia gama de productos y servicios.
    - **Impacto:** Ha democratizado el acceso a la tecnología móvil, permitiendo la proliferación de dispositivos asequibles y fomentando un ecosistema vibrante de aplicaciones móviles.

- **OpenStack:**

- **Caso de éxito:** Es una plataforma de infraestructura en la nube, utilizada por empresas como IBM, NASA y PayPal. Permite construir y gestionar infraestructuras en la nube de manera eficiente y escalable.
    - **Impacto:** Ha permitido a las empresas adoptar la computación en la nube de manera más flexible y económica, impulsando la innovación y la eficiencia operativa.

# Software Libre.

- **Definición.**

- Es un tipo de software que **respeta la libertad de los usuarios** y promueve la colaboración y la comunidad.
- "Libre" y "gratis" (en inglés, "free" y "free of charge") **no son equivalentes**.
- El software libre **puede ser gratuito o tener un costo**. Lo esencial es que los usuarios tienen libertad.
- La **Free Software Foundation (FSF)**, fundada por **Richard Stallman**, define el software libre como el **software que otorga a los usuarios cuatro libertades esenciales**.

# Software Libre.

- **Las Cuatro Libertades del Software Libre.**

- **Libertad 0: La libertad de usar el programa con cualquier propósito**
  - Cualquier persona pueda ejecutar el software para cualquier propósito sin restricciones.
- **Libertad 1: La libertad de estudiar cómo funciona el programa y cambiarlo para que haga lo que el usuario quiera**
  - El acceso al código fuente es una condición previa. Permite a los usuarios comprender el funcionamiento interno del software y adaptarlo a sus necesidades específicas.
- **Libertad 2: La libertad de redistribuir copias para ayudar a otros**
  - Cualquier persona pueda compartir el software con otros, gratis o con un costo. Fomenta la colaboración y la distribución del software.
- **Libertad 3: La libertad de distribuir copias de sus versiones modificadas a otros**
  - Permite distribuir versiones modificadas del software a otras personas, garantizando que la comunidad pueda beneficiarse de las contribuciones individuales.



# Software Libre.

## • Importancia de las cuatro libertades.

### • Promoción de la colaboración:

- Modificar y redistribuir el software fomentan una comunidad activa de desarrolladores y usuarios que colaboran para mejorar el software.

### • Aseguramiento de la transparencia:

- El acceso al código y poder estudiarlo aseguran que el software sea transparente, permitiendo auditorías independientes para identificar y corregir vulnerabilidades.

### • Fomento de la independencia:

- Usar y modificar el software garantizan la no dependencia de un solo proveedor para soporte y actualizaciones.

### • Acceso universal:

- Redistribuir copias asegura que el software esté disponible para una amplia audiencia, independientemente de las barreras económicas.

# Historia y Filosofía del Movimiento del Software Libre.

## • Origen y evolución del Movimiento del Software Libre.

### • Inicios y Cultura Hacker:

- En los años 60 y 70, los programadores compartían el código de los programas.
- Los "**hackers**", trabajaban para mejorar el sw y resolver problemas técnicos.

### • Surgimiento del software propietario:

- A finales de los años 70 y principios de los 80, las empresas empezaron a vender sw, restringiendo el acceso al código, su redistribución y modificación.
- Ejemplos: Unix de AT&T, que se convirtió en propietario, y el software de Microsoft.

### • Lanzamiento del Proyecto GNU:

- En 1983, **Richard Stallman**, un programador del MIT, anunció el Proyecto GNU (GNU's Not Unix) con el objetivo de desarrollar un SO completamente libre.
- Stallman se inspiró en la necesidad de crear software que respetara la libertad de los usuarios y que promoviera la colaboración abierta.

## Historia y Filosofía del Movimiento del Software Libre.

### • Origen y evolución del Movimiento del Software Libre.

#### • **Fundación de la Free Software Foundation (FSF):**

- En 1985, Stallman fundó la Free Software Foundation (FSF).
- Se convirtió en la principal organización que defiende y promueve el software libre.

#### • **Desarrollo del Núcleo Linux:**

- En 1991 Linus Torvalds desarrolló el núcleo Linux. Al combinarse con el Proyecto GNU, se creó un SO completo conocido como GNU/Linux.

#### • **Expansión y adopción del Software Libre:**

- Durante los 90's y 2000, el SL comenzó a ganar popularidad. Proyectos como Apache, Mozilla Firefox y LibreOffice se convirtieron en ejemplos prominentes de software libre.
- La adopción de GNU/Linux en servidores, móviles y supercomputadoras contribuyó significativamente al crecimiento y aceptación del software libre en la industria.

## Historia y Filosofía del Movimiento del Software Libre.

### • Principios y valores promovidos por el Movimiento.

- **Ética y libertad:** Principios éticos que valoran la libertad del usuario.
- **Las cuatro libertades:** Usar, estudiar, redistribuir y modificar.
- **Rechazo del software propietario:** El software propietario restringe las libertades de los usuarios, impidiendo la colaboración y el progreso tecnológico.
- **Licencias de software libre:** Aseguran que cualquier software derivado de un software libre también deba ser libre.
- **Comunidad y colaboración:** Trabajo conjunto para mejorar el software, compartir conocimientos y resolver problemas.
- **Impacto social:** Promover el acceso universal a la tecnología, fomentar la educación y el aprendizaje, y defender los derechos de los usuarios a través de la transparencia y la colaboración.

## Historia y Filosofía del Movimiento del Software Libre.

- **El Papel de la Free Software Foundation (FSF) y del Proyecto GNU.**

- **Free Software Foundation (FSF):**

- **Misión:** Asegurar que los usuarios puedan controlar la tecnología que utilizan.
    - **Actividades:** Desarrolla y mantiene licencias de software libre, y proporciona recursos y apoyo a proyectos de software libre.
    - **Defensa y educación:** Defiende los derechos de los usuarios de software y educa al público sobre la importancia del software libre.

- **Proyecto GNU:**

- **Objetivo:** Desarrollar un SO completo y libre, conocido como GNU. El nombre "GNU" significa "GNU's Not Unix", reflejando su intención de ser una alternativa libre a Unix.
    - **Componentes:** El Proyecto GNU ha desarrollado una amplia gama de herramientas y utilidades de software, incluyendo el compilador GCC, el editor de texto Emacs y muchas otras.
    - **Colaboración con Linux:** El núcleo original de GNU (Hurd) no llegó a ser completamente funcional, el núcleo Linux se combinó con GNU para formar el SO GNU/Linux.

## Licencias de Software Libre.

- **Introducción.**

- Las licencias de software libre son fundamentales para garantizar las libertades que definen el software libre.
  - Existen varios tipos de licencias, cada una con sus propias condiciones y requisitos.
  - Estas licencias se pueden agrupar en dos categorías principales:
    - Licencias permisivas.
    - Licencias copyleft.

## Licencias de Software Libre.

- **Diferencias entre licencias permisivas y copyleft.**

- **Licencias copyleft:**

- **Definición:** Aseguran que cualquier software derivado de un programa licenciado bajo copyleft también debe ser libre y distribuido bajo los mismos términos.
    - **Propósito:** Mantener la libertad del software a lo largo de toda su distribución y modificación, evitando que se convierta en software propietario.
    - **Ejemplo principal:** **GPL** (Licencia Pública General de GNU).

- **Licencias permisivas:**

- **Definición:** Permiten un uso más flexible del software, incluyendo la integración en software propietario.
    - **Propósito:** Permiten utilizar el software con menos restricciones, lo que puede facilitar una adopción más amplia y una mayor integración con software propietario.
    - **Ejemplos principales:** **MIT**, **Apache**, **BSD**.

## Licencias de Software Libre.

- **Tipo de licencias.**

- **Licencia Pública General de GNU (GPL):**

- **Descripción:** Es una de las licencias de software libre más conocidas y utilizadas. Fue creada por Richard Stallman y la Free Software Foundation (FSF) para el Proyecto GNU.
    - **Condiciones:** Asegura que cualquier software derivado de un programa licenciado bajo la GPL también debe ser libre y estar disponible bajo los mismos términos. Obliga a que el código fuente esté disponible y permite modificar y redistribuir el software bajo la misma licencia.
    - **Versión actual:** La versión más reciente es la GPLv3, que aborda problemas legales y técnicos que surgieron desde la publicación de la versión anterior.

## Licencias de Software Libre.

- **Tipo de licencias.**

- **Licencia Pública General Reducida de GNU (LGPL):**

- **Descripción:** Es una variante más permisiva de la GPL, diseñada principalmente para bibliotecas de software.
    - **Condiciones:** Permite que las bibliotecas se utilicen en programas propietarios, siempre y cuando las modificaciones se distribuyan bajo la LGPL.

- **Licencia MIT (Massachusetts Institute of Technology):**

- **Descripción:** Es una de las licencias permisivas más simples y flexibles.
    - **Condiciones:** Permite a los usuarios hacer prácticamente cualquier cosa con el software, incluyendo el uso, copia, modificación, fusión, publicación, distribución, sublicencia y venta del software, siempre y cuando se incluya el aviso de derechos de autor original y la renuncia de responsabilidad.

## Licencias de Software Libre.

- **Tipo de licencias.**

- **Licencia Apache:**

- **Descripción:** Es otra licencia permisiva ampliamente utilizada, mantenida por la Apache Software Foundation.
    - **Condiciones:** Permite uso, modificación y distribución del software bajo condiciones similares a la licencia MIT, pero también incluye disposiciones explícitas sobre patentes y la necesidad de reconocer cambios en los archivos originales.

- **Licencia BSD (Berkeley Software Distribution):**

- **Descripción:** Es otra licencia permisiva que proviene del sistema operativo BSD.
    - **Condiciones:** Similar a la MIT, permite el uso, modificación y distribución del software, pero con una cláusula adicional que prohíbe el uso del nombre de los desarrolladores originales para promocionar productos derivados sin permiso.



## Licencias de Software Libre.

- **Ejemplos de proyectos bajo diferentes licencias.**

- **Proyectos bajo GPL:**

- **Linux Kernel:** El núcleo de Linux es uno de los ejemplos más conocidos de software licenciado bajo la GPL.
    - **GNU Compiler Collection (GCC):** Una colección de compiladores desarrollada por el Proyecto GNU.

- **Proyectos bajo LGPL:**

- **GNU C Library (glibc):** La biblioteca estándar de C para sistemas operativos GNU.
    - **FFmpeg:** Un conjunto de bibliotecas y programas para manejar datos multimedia.

- **Proyectos bajo MIT:**

- **jQuery:** Una popular biblioteca de JavaScript.
    - **Ruby on Rails:** Un framework para aplicaciones web escrito en Ruby.

## Licencias de Software Libre.

- **Ejemplos de proyectos bajo diferentes licencias.**

- **Proyectos bajo Apache:**

- **Apache HTTP Server:** Uno de los servidores web más utilizados en el mundo.
    - **Apache Hadoop:** Un marco de software para el procesamiento distribuido de grandes conjuntos de datos.

- **Proyectos bajo BSD:**

- **FreeBSD:** Un sistema operativo derivado de BSD.
    - **OpenSSH:** Un conjunto de herramientas para la conexión segura a sistemas remotos.



## Comparación con Software Propietario.

### • Diferencias entre Software Libre y Software Propietario.

#### • Acceso al código fuente:

- **Software Libre:** El código fuente está disponible para todos.
- **Software Propietario:** Solo el creador del software o personas autorizadas pueden acceder y modificar el código.

#### • Libertades del usuario:

- **Software Libre:** Libertad de usar, copiar, modificar y distribuir el software.
- **Software Propietario:** Los usuarios están restringidos en cómo pueden usar el software.

#### • Licenciamiento:

- **Software Libre:** Bajo términos que promueven la libertad de uso y modificación.
- **Software Propietario:** Licenciado bajo términos restrictivos que limitan el uso, la copia y la modificación del software.

## Comparación con Software Propietario.

### • Diferencias entre Software Libre y Software Propietario.

#### • Desarrollo y colaboración:

- **Software Libre:** Desarrollado de manera colaborativa por comunidades de desarrolladores y usuarios de todo el mundo.
- **Software Propietario:** Desarrollado por equipos cerrados de desarrolladores dentro de una empresa o entidad.

#### • Costos:

- **Software Libre:** Generalmente gratuito, aunque puede haber costos asociados con el soporte y los servicios adicionales.
- **Software Propietario:** Usualmente requiere la compra de una licencia, suscripciones o pagos periódicos.

## Comparación con Software Propietario.

- **Ventajas y desventajas de cada enfoque.**

- **Software Libre:**

- **Ventajas:**

- **Transparencia:** El acceso al código fuente permite a los usuarios verificar lo que hace el software, garantizando mayor seguridad y confianza.
      - **Flexibilidad:** Los usuarios pueden modificar el software para adaptarlo a sus necesidades específicas.
      - **Costos:** Generalmente no requiere costos de licencia, lo que puede ser más económico para organizaciones y usuarios individuales.
      - **Comunidad y soporte:** Una comunidad activa puede proporcionar soporte, actualizaciones y mejoras continuas.

## Comparación con Software Propietario.

- **Ventajas y desventajas de cada enfoque.**

- **Software Libre:**

- **Desventajas:**

- **Soporte profesional:** Puede haber una falta de soporte profesional dedicado, aunque muchas empresas ofrecen soporte para software libre.
      - **Interfaz y usabilidad:** Algunas aplicaciones de software libre pueden tener interfaces menos pulidas en comparación con sus contrapartes propietarias.
      - **Compatibilidad:** Puede haber problemas de compatibilidad con otros sistemas propietarios o estándares cerrados.

## Comparación con Software Propietario.

- **Ventajas y desventajas de cada enfoque.**

- **Software Propietario:**

- **Ventajas:**
    - **Soporte profesional:** Generalmente incluye soporte profesional dedicado, incluyendo asistencia técnica y actualizaciones regulares.
    - **Interfaz y usabilidad:** Suelen tener interfaces de usuario más refinadas y amigables, diseñadas para atraer a una amplia audiencia.
    - **Integración:** A menudo está optimizado para integrarse bien con otros productos de la misma empresa o ecosistema.

## Comparación con Software Propietario.

- **Ventajas y desventajas de cada enfoque.**

- **Software Propietario:**

- **Desventajas:**
    - **Costos:** Puede ser costoso debido a los precios de las licencias, suscripciones y actualizaciones.
    - **Falta de flexibilidad:** Los usuarios no pueden modificar el software para adaptarlo a sus necesidades específicas.
    - **Dependencia del proveedor:** Los usuarios dependen del proveedor para las actualizaciones y correcciones de errores, lo que puede ser problemático si el soporte se discontinúa.

## Comparación con Software Propietario.

- **Impacto en la industria y en los usuarios.**

- **Impacto en la industria:**

- **Innovación y desarrollo:** Colaboración abierta para mejora continua.
    - **Competencia y mercado:** Costos reducidos y maximización de la competitividad.
    - **Adopción por grandes empresas:** Contribución y aumento de la credibilidad.

- **Impacto en los usuarios:**

- **Acceso y participación:** Acceso sin costo y participación en el desarrollo y mejora.
    - **Control y personalización:** Adaptación y control de cómo se utiliza.
    - **Seguridad y privacidad:** Transparencia para mayor seguridad y privacidad.
    - **Costos reducidos:** Beneficio para organizaciones sin fines de lucro, instituciones educativas y usuarios individuales.

**Fin del Módulo I**