Juan Menduiña

Trabajo Práctico N° 1:
GNU/Linux, Instalación y Conceptos Básicos, Permisos, Arranque, Usuarios. Organización Interna.

Ejercicio 1: Características de GNU/Linux.

(a) *Mencionar y explicar las características más relevantes de GNU/Linux.*

GNU/Linux es un sistema operativo libre y de código abierto, basado en el núcleo Linux y en las herramientas del proyecto GNU. Sus características más relevantes son:

1. ***Software* libre y código abierto:**
   GNU/Linux se distribuye bajo licencias libres (como la GPL), lo que permite a los usuarios usar, estudiar, modificar y redistribuir el *software*. Esta libertad fomenta la colaboración y la mejora continua del sistema.
2. **Multitarea y multiusuario:**
   Permite que varios usuarios trabajen, simultáneamente, en el mismo sistema sin interferir entre sí y que múltiples procesos se ejecuten al mismo tiempo. Esto lo hace ideal para entornos de servidores y redes.
3. **Portabilidad:**
   GNU/Linux puede ejecutarse en una amplia variedad de plataformas y arquitecturas (x86, ARM, RISC-V, etc.), desde servidores y computadoras personales hasta dispositivos embebidos y teléfonos.
4. **Seguridad y estabilidad:**
   Es reconocido por su gran estabilidad, incluso en ejecuciones prolongadas, y por su robusto sistema de permisos y usuarios, que limita el alcance de posibles ataques o errores. Además, la comunidad realiza actualizaciones y parches de seguridad constantemente.
5. **Estructura modular:**
   Está compuesto por módulos independientes (núcleo, *shell*, sistema de archivos, utilidades, etc.), lo que facilita su mantenimiento y personalización.
6. **Sistema de archivos jerárquico:**
   Organiza todos los recursos (archivos, dispositivos, configuraciones) en una única estructura jerárquica, que parte del directorio raíz (/).
7. **Compatibilidad y flexibilidad:**
   Existen numerosas distribuciones (Ubuntu, Debian, Fedora, Arch, etc.) adaptadas a distintos tipos de usuarios y necesidades. Además, permite elegir entre múltiples entornos de escritorio, gestores de paquetes y herramientas.
8. **Comunidad y soporte colaborativo:**
   GNU/Linux cuenta con una gran comunidad internacional que desarrolla, documenta y brinda soporte de manera colaborativa.

(b) *Mencionar otros sistemas operativos y compararlos con GNU/Linux en cuanto a los puntos mencionados en el inciso (a).*

Otros sistemas operativos, ampliamente utilizados, son Windows (Microsoft), masOS (Apple) y Android (Google). A continuación, se comparan con GNU/Linux en cuanto a los puntos mencionados en el inciso (a):

1. ***Software* libre y código abierto:**
   - Windows: Es propietario y cerrado; el código fuente no está disponible y su uso está sujeto a licencias comerciales.
   - macOS: Es privativo y exclusivo del *hardware* de Apple, aunque incorpora componentes de código abierto (como partes de BSD).
   - Android: Parcialmente abierto, ya que el núcleo es Linux, pero muchas capas y servicios de Google son propietarios.
2. **Multitarea y multiusuario:**
   Todos los sistemas mencionados permiten multitarea (ejecución simultánea de procesos). Sin embargo, GNU/Linux fue diseñado desde sus orígenes como multiusuario real, algo que, en Windows y macOS, se implementó más tarde y con mayores restricciones.
3. **Portabilidad:**
   - Windows: Está optimizado para arquitecturas x86/x64; su portabilidad es limitada.
   - macOS: Funciona, exclusivamente, en equipos Apple.
   - Android: Muy extendido en dispositivos móviles, pero no tan flexible fuera de ese entorno.
4. **Seguridad y estabilidad:**
   - Windows: Más vulnerable a virus y *malware*, principalmente por su popularidad y arquitectura de permisos más laxa.
   - *macOS:* Tiene buena seguridad, aunque no alcanza la flexibilidad ni el control de GNU/Linux.
   - Android: Depende de las capas del fabricante y de la actualización; puede presentar vulnerabilidades si no se actualiza.
5. **Estructura modular:**
   - GNU/Linux: Altamente modular; el usuario puede modificar o reemplazar componentes del sistema.
   - Windows y macOS: Mucho más cerrados; el usuario tiene escaso control sobre los módulos del sistema.
   - Android: Parcialmente modular, pero condicionado por Google y los fabricantes.
6. **Sistema de archivos jerárquico:**
   Todos utilizan una estructura jerárquica, aunque en GNU/Linux es más uniforme (todo se organiza bajo "/"). En Windows, existen múltiples unidades (C:, D:, etc.), lo que fragmenta la estructura.
7. **Compatibilidad y flexibilidad:**
   - Windows: Ampliamente compatible con programas comerciales y *hardware*, pero poco flexible; el usuario no puede modificar ni personalizar, en profundidad, el sistema.
   - macOS: Ofrece buena compatibilidad dentro del ecosistema Apple y gran estabilidad, pero es cerrado y limitado al *hardware* de la marca.
   - Android: Posee gran compatibilidad con aplicaciones móviles, pero su flexibilidad depende del fabricante y de las capas de *software* agregadas, que restringen la personalización completa.

8. **Comunidad y soporte:**
   - Windows y macOS: El soporte depende de las empresas propietarias.
   - Android: Tiene soporte de Google y comunidades específicas, pero menos abiertas que en GNU/Linux.

En síntesis, GNU/Linux se destaca por su libertad, seguridad, estabilidad y flexibilidad, mientras que los sistemas operativos propietarios como Windows y macOS ofrecen mayor facilidad de uso y compatibilidad comercial, pero a costa de menor control por parte del usuario.

(c) *¿Qué es GNU?*

GNU es un proyecto de *software* libre iniciado en 1983 por Richard Stallman, con el objetivo de crear un sistema operativo completamente libre, compatible con Unix, pero sin incluir ningún componente privativo. El nombre GNU es un acrónimo recursivo que significa "GNU's Not Unix" ("GNU No es Unix").

El proyecto forma parte del movimiento del *software* libre, que defiende las cuatro libertades fundamentales del usuario:

- Usar el programa con cualquier propósito.
- Estudiar cómo funciona y adaptarlo a las necesidades propias.
- Redistribuir copias.
- Mejorar el programa y publicar esas mejoras.

El sistema GNU proporciona las herramientas básicas de un sistema operativo (compiladores, bibliotecas, *shells*, utilidades de administración, editores de texto, etc.).

En la actualidad, la mayoría de las distribuciones conocidas como "Linux" son, en realidad, combinaciones del núcleo Linux con las herramientas del proyecto GNU, motivo por el cual su nombre correcto es GNU/Linux.

(d) *Indicar una breve historia sobre la evolución del proyecto GNU.*

El proyecto GNU fue iniciado en 1983 por Richard Stallman en el Instituto Tecnológico de Massachusetts (MIT), con la idea de desarrollar un sistema operativo completamente libre y compatible con Unix.

En 1985, Stallman fundó la *Free Software Foundation* (FSF) para apoyar el desarrollo y la difusión del *software* libre, así como para promover licencias que garantizaran la libertad de los usuarios, como la *General Public Licence* de GNU (GPL).

Durante la segunda mitad de la década de 1980, el proyecto GNU avanzó en la creación de herramientas esenciales como el compilador GCC (GNU *Compiler Collection*), el editor Emacs, el intérprete de comandos Bash y muchas utilidades del sistema. Sin

embargo, el sistema GNU aún no contaba con un núcleo funcional (llamado Hurd), que se encontraba en desarrollo.

En 1991, Linus Torvalds publicó el núcleo Linux, que era libre y compatible con las herramientas GNU. La combinación de ambos permitió conformar un sistema operativo completo: GNU/Linux, el cual se difundió, rápidamente, en universidades, empresas y entornos domésticos.

Desde entonces, el proyecto GNU continúa activo, manteniendo y desarrollando múltiples programas libres, y promoviendo los principios éticos y sociales del movimiento del *software* libre.

(e) *Explicar qué es la multitarea e indicar si GNU/Linux hace uso de ella.*

La multitarea es la capacidad de un sistema operativo para ejecutar varios procesos o programas de manera simultánea, compartiendo los recursos del procesador, la memoria y otros dispositivos.

En realidad, el procesador alterna, rápidamente, entre las distintas tareas, dando la sensación de ejecución paralela (especialmente, en sistemas con un solo núcleo). En los procesadores multinúcleo, varias tareas pueden ejecutarse, verdaderamente, al mismo tiempo.

Existen dos tipos principales de multitarea:

- Cooperativa: Cada proceso cede, voluntariamente, el control al sistema operativo.
- Preventiva: El sistema operativo decide cuándo interrumpir un proceso para dar tiempo de CPU a otro.

GNU/Linux implementa multitarea preventiva, lo que significa que el núcleo administra, de forma automática, el uso del procesador entre los distintos procesos. Esto permite que el sistema siga funcionando de manera fluida, incluso cuando muchos programas están activos al mismo tiempo.

(f) *¿Qué es POSIX?*

POSIX (siglas de *Portable Operating System Interface*, o "Interfaz Portátil para Sistemas Operativos") es un conjunto de estándares definidos por IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) que especifica una interfaz común para los sistemas operativos tipo Unix.

El objetivo de POSIX es garantizar que los programas puedan compilarse y ejecutarse en distintos sistemas operativos compatibles, sin necesidad de modificar su código fuente. Para lograrlo, define normas sobre aspectos como:

- Llamadas al sistema (*system calls*).

- Estructura del sistema de archivos.
- Manejo de procesos e hilos.
- Señales y comunicación entre procesos.
- Comportamiento del *shell* y utilidades básicas.

GNU/Linux cumple, en gran medida, con el estándar POSIX, lo que permite que muchas aplicaciones desarrolladas para sistemas Unix (como BSD o macOS) puedan ejecutarse en Linux con muy pocas modificaciones.

En resumen, POSIX promueve la portabilidad y la compatibilidad entre los distintos sistemas operativos de la familia Unix y sus derivados.

Ejercicio 2: Distribuciones de GNU/Linux.

(a) *¿Qué es una distribución de GNU/Linux? Nombrar, al menos, 4 distribuciones de GNU/Linux y citar diferencias básicas entre ellas.*

Una distribución de GNU/Linux es un paquete completo de *software* que combina el núcleo Linux, las herramientas del proyecto GNU y otros programas adicionales (como gestores de paquetes, entornos de escritorio y aplicaciones) para ofrecer un sistema operativo listo para instalar y usar. Cada distribución se adapta a distintos propósitos, como servidores, escritorios, educación o seguridad informática.

Algunas distribuciones populares son:

1. **Ubuntu:**
   - Orientada a usuarios de escritorio y principiantes.
   - Fácil instalación y amplia comunidad de soporte.
   - Basada en Debian, con un ciclo de actualizaciones regular y soporte a largo plazo (LTS).
2. **Debian:**
   - Enfocada en estabilidad y robustez.
   - Ideal para servidores y entornos críticos.
   - Sus paquetes son más conservadores; menos actualizaciones frecuentes.
3. **Fedora:**
   - Distribución vanguardista, incluye *software* más reciente.
   - Buena para desarrolladores y pruebas de nuevas tecnologías.
   - Respaldo de *Red Hat*, aunque con un ciclo de vida más corto.
4. **Arch Linux:**
   - Minimalista y altamente personalizable.
   - Filosofía "*rolling release*" (actualizaciones continuas).
   - Requiere mayor conocimiento técnico para su instalación y mantenimiento.

En resumen, cada distribución combina los mismos elementos básicos de GNU/Linux, pero se diferencian en facilidad de uso, frecuencia de actualizaciones, estabilidad y personalización.

(b) *¿En qué se diferencia una distribución de otra?*

Aunque todas las distribuciones de GNU/Linux comparten el núcleo Linux y las herramientas GNU, se diferencian, principalmente, en los siguientes aspectos:

1. **Gestor de paquetes:**
   Cada distribución utiliza su propio sistema para instalar y actualizar *software*:
   - Debian/Ubuntu usan APT y paquetes .deb.
   - Fedora/Red Hat usan DNF/YUM y paquetes .rpm.
   - Arch Linux usa Pacman.
2. **Ciclo de actualizaciones y estabilidad:**

- Algunas distribuciones (Debian Stable, Ubuntu LTS) priorizan estabilidad, con versiones de *software* más conservadoras.
- Otras (Fedora, Arch Linux) priorizan lo último en *software*, con actualizaciones frecuentes.

3. **Facilidad de uso e instalación:**
   - Distribuciones como Ubuntu están orientadas a usuarios principiantes, con instaladores gráficos y entornos de escritorio listos para usar.
   - Arch Linux o Gentoo requieren conocimientos avanzados, con instalación y configuración manual.
4. **Entorno de escritorio y personalización:**
   - Algunas distribuciones vienen con entornos gráficos predeterminados (GNOME, KDE, XFCE).
   - Otras permiten al usuario elegir o instalar el entorno que prefiera.
5. **Propósito y enfoque:**
   - Algunas distribuciones se enfocan en servidores (Debian, CentOS), otras en escritorio general (Ubuntu, Linux Mint) y otras en seguridad o pruebas de penetración (Kali Linux, Parrot OS).

En resumen, las diferencias entre distribuciones se centran en gestión de paquetes, frecuencia de actualizaciones, facilidad de uso, entornos gráficos y finalidad del sistema, mientras que el núcleo Linux y las herramientas GNU siguen siendo comunes a todas.

(c) *¿Qué es Debian? Acceder al sitio https://www.debian.org/ e indicar cuáles son los objetivos del proyecto y una breve cronología del mismo.*

Debian es un sistema operativo libre y universal, compuesto por el núcleo Linux y herramientas del proyecto GNU. Es mantenido por una comunidad internacional de desarrolladores y usuarios comprometidos con el *software* libre. Su nombre proviene de la combinación de los nombres de sus creadores: Debra Lynn y Ian Murdock.

Objetivos del proyecto: Según su filosofía, Debian busca crear un sistema operativo libre, disponible para todo el mundo, sin importar su ubicación geográfica, idioma o nivel de conocimiento técnico. El proyecto se basa en principios de apertura, transparencia y colaboración comunitaria. No se enfoca en el beneficio económico, sino en el desarrollo ético y técnico del *software* libre.

Cronología del proyecto Debian:

- Agosto de 1993: Ian Murdock, estudiante de la Universidad de Purdue, inicia el proyecto Debian con el objetivo de crear una distribución de Linux completamente libre.
- 1994: Se publica la versión 0.91, que incluye un sistema de empaquetado básico y una estructura inicial de paquetes.
- 1996: Debian 1.1, conocida como “Buzz”, es lanzada, marcando la primera versión oficial del sistema.
- 1999: Se inicia el soporte para arquitecturas adicionales y se publica la versión 2.0, conocida como “Hamm”.

- 2005: Debian 3.1, conocida como “Sarge”, es lanzada después de un largo período de desarrollo.
- 2015: Se implementa el soporte a largo plazo (LTS) para versiones antiguas, extendiendo su vida útil y seguridad.
- 2025: Debian 13, conocida como “Trixie”, es lanzada, continuando con el compromiso de ofrecer un sistema operativo libre, estable y seguro.

Esta cronología destaca los hitos importantes en el desarrollo y evolución del proyecto Debian, reflejando su crecimiento y consolidación como una de las distribuciones más respetadas y utilizadas en el mundo del *software* libre.

Ejercicio 3: Estructura de GNU/Linux.

(a) *Nombrar cuáles son los 3 componentes fundamentales de GNU/Linux.*

Los tres componentes fundamentales de GNU/Linux son:

1. **Núcleo (*Kernel*):**
   - Es el corazón del sistema operativo.
   - Se encarga de gestionar el *hardware*, la memoria, los procesos y la comunicación entre dispositivos y programas.
   - Ejemplos: Linux kernel 6.x, 5.x, etc.
2. ***Shell* o intérprete de comandos:**
   - Es la interfaz entre el usuario y el núcleo.
   - Permite ejecutar comandos, *scripts* y programas.
   - Ejemplos: Bash, Zsh, Fish.
3. **Sistema de archivos y utilidades GNU:**
   - Conjunto de herramientas y programas esenciales para operar el sistema, como compiladores, editores de texto, utilidades de gestión de archivos y bibliotecas.
   - Incluye la estructura jerárquica de directorios, desde la raíz (/) hasta las carpetas de configuración y datos de usuario.

(b) *Mencionar y explicar la estructura básica del Sistema Operativo GNU/Linux.*

El sistema operativo GNU/Linux se organiza en varias capas jerárquicas, que permiten separar las funciones y facilitan la gestión del sistema. La estructura básica es la siguiente:

1. **Núcleo (*Kernel*):**
   - Es la capa más interna, responsable de la comunicación entre el *hardware* y el *software*.
   - Gestiona la CPU, memoria, dispositivos de entrada/salida, procesos y control de acceso.
   - Funciona como intermediario entre los programas y los recursos físicos del sistema.
2. ***Shell* o intérprete de comandos:**
   - Es la capa intermedia que permite al usuario interactuar con el sistema.
   - Puede ser texto (CLI - *Command Line Interface*) o gráfica (GUI), dependiendo de la configuración.
   - A través del *shell*, se pueden ejecutar comandos, *scripts* y programas, controlar procesos y administrar archivos.
3. **Sistema de archivos y utilidades:**
   - Incluye las herramientas básicas de GNU y otros programas esenciales (compiladores, editores, utilidades de red, bibliotecas, etc.)
   - Organiza los archivos en una estructura jerárquica que comienza en la raíz (/) y se ramifica en directorios como:
     - */bin*: programas esenciales.

- */etc*: archivos de configuración.
- */home*: directorios de usuarios.
- */usr*: programas y utilidades adicionales.
- */var*: archivos variables (logs, bases de datos).

4. **Aplicaciones y entorno de escritorio:**
   - Son los programas que el usuario final utiliza, como navegadores, procesadores de texto, reproductores de multimedia o entornos gráficos (GNOME, KDE, XFCE).
   - Esta capa depende de las anteriores para funcionar, ya que utiliza los servicios del *kernel*, el *shell* y las bibliotecas del sistema.

En resumen, GNU/Linux tiene una estructura modular y jerárquica, *Kernel* ⟶ *Shell* ⟶ Sistema de archivos/utilidades ⟶ Aplicaciones, lo que permite seguridad, estabilidad y flexibilidad en la gestión del sistema.

Ejercicio 4: *Kernel*.

(a) *¿Cuáles son sus funciones principales?*

El *kernel* es el núcleo del sistema operativo y su función principal es actuar como intermediario entre el *hardware* y el *software*. Sus funciones más importantes son:

1. **Gestión de procesos:**
   - Controla la creación, la planificación y la terminación de los procesos en ejecución.
   - Implementa la multitarea y asigna tiempo de CPU a cada proceso de manera eficiente.
2. **Gestión de memoria:**
   - Administra la memoria principal (RAM) y el espacio de intercambio (*swap*).
   - Se encarga de asignar memoria a los procesos y de proteger áreas de memoria para que un proceso no interfiera con otro.
3. **Gestión de dispositivos:**
   - Controla todos los dispositivos de *hardware* mediante los *drivers*.
   - Permite que los programas accedan a los recursos de forma estandarizada y segura.
4. **Gestión del sistema de archivos:**
   - Maneja la lectura, la escritura y la organización de los datos en los discos y otros medios de almacenamiento.
   - Garantiza la integridad de la información y permite acceso jerárquico a los archivos.
5. **Gestión de la seguridad y permisos:**
   - Controla los permisos de usuario y el acceso a recursos, evitando que procesos no autorizados realicen acciones críticas.
   - Implementa mecanismos de protección de memoria y de ejecución de código.
6. **Comunicación entre procesos:**
   - Facilita la interacción y la sincronización entre procesos mediante señales, tuberías, *sockets* y memoria compartida.

En resumen, el *kernel* es el cerebro del sistema, responsable de que los procesos, la memoria, el *hardware* y los archivos funcionen de manera coordinada y segura, garantizando la estabilidad y la eficiencia del sistema operativo.

(b) *¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado del Kernel en versiones anteriores a la 2.4? ¿Qué cambió en el versionado que se impuso a partir de la versión 2.6?*

Versión actual del kernel: Hasta el 16 de octubre de 2025, la versión estable más reciente del *kernel* de Linux es la 6.17.3. Esta versión se lanzó el 15 de octubre de 2025 y es la que se utiliza en distribuciones como Ubuntu 25.10 y Fedora 43.

Esquema de versionado antes de la versión 2.4: Antes de la versión 2.4, el esquema de versionado del *kernel* de Linux seguía una convención basada en números impares y pares:

- Números impares (por ejemplo, 2.1, 2.3, 2.5): Indicaban versiones en desarrollo, con cambios experimentales y no estables.
- Números pares (por ejemplo, 2.0, 2.2): Representaban versiones estables, listas para su uso general.

Este enfoque permitía a los desarrolladores y usuarios identificar, rápidamente, si una versión era estable o aún estaba en desarrollo.

Cambios en el esquema de versionado a partir de la versión 2.6: Con la introducción de la versión 2.6, el esquema de versionado cambió para reflejar mejor el ciclo de desarrollo y mantenimiento del *kernel*:

- Número mayor (2): Indicaba la serie principal del kernel.
- Número menor (6): Representaba la versión de desarrollo.
- Número de revisión (x): Denotaba actualizaciones menores o correcciones de errores.
- Número de parche (y): Se añadía para indicar parches específicos o actualizaciones de seguridad.

Este nuevo esquema proporcionaba una estructura más clara y coherente para el seguimiento de las versiones del *kernel* y facilitaba la gestión de actualizaciones y mantenimiento.

(c) *¿Es posible tener más de un Kernel de GNU/Linux instalado en la misma máquina?*

Sí, es posible tener más de un *kernel* de GNU/Linux instalado en la misma máquina. Esto se logra porque cada versión del *kernel* se instala en su propio directorio dentro de /boot y se registra en el GRUB (el gestor de arranque), que permite seleccionar qué *kernel* iniciar al arrancar el sistema.

Ventajas de tener múltiples *kernels*:

- Seguridad ante fallos: Si la versión más reciente del *kernel* causa problemas, se puede iniciar el sistema con una versión anterior que funcione correctamente.
- Compatibilidad: Permite probar nuevas versiones del *kernel* sin comprometer la estabilidad del sistema principal.
- Flexibilidad para desarrolladores: Facilita el desarrollo y las pruebas de módulos o *drivers* específicos para distintas versiones del *kernel*.

En resumen, tener más de un *kernel* instalado es común y seguro, y permite mantener la estabilidad del sistema mientras se prueban actualizaciones o nuevas características.

(d) *¿Dónde se encuentra ubicado dentro del File System?*

En GNU/Linux, el *kernel* no es un programa como cualquier otro, es un archivo especial que se carga en memoria al arrancar el sistema. Dentro del *File System*, se encuentra ubicado en el directorio */boot*, que contiene los archivos del *kernel* instalados en el sistema.

Ejercicio 5: Intérprete de comandos (*Shell*).

(a) *¿Qué es?*

El *shell* es un programa que actúa como intermediario entre el usuario y el núcleo (*kernel*) del sistema operativo. Su función principal es interpretar los comandos que ingresa el usuario y comunicarlos al *kernel* para que se ejecuten.

El *shell* es la interfaz de línea de comandos que permite a los usuarios interactuar con GNU/Linux de manera flexible y poderosa, tanto de forma interactiva como mediante *scripts*.

(b) *¿Cuáles son sus funciones?*

Sus funciones son:

- Interpretar comandos: Permite ejecutar programas, *scripts* o instrucciones del sistema.
- Automatización de tareas: A través de *scripts*, se pueden automatizar procesos repetitivos.
- Gestión de procesos: Permite iniciar, detener o supervisar procesos.
- Interacción con el sistema de archivos: Navegar por directorios, copiar, mover o eliminar archivos.
- Variables y programación básica: Permite usar variables, condicionales y bucles para tareas más complejas.

(c) *Mencionar, al menos, 3 intérpretes de comandos que posee GNU/Linux y compararlos entre ellos.*

GNU/Linux ofrece varios intérpretes de comandos (*shells*), cada uno con características particulares. A continuación, se mencionan 3 de los más utilizados y una breve comparación entre ellos:

1. **Bash (*Bourne Again Shell*):**
   - Es el *shell* por defecto en la mayoría de las distribuciones GNU/Linux.
   - Ventajas: compatible con *scripts* del *shell* original (sh); soporta historial de comandos, autocompletado y variables de entorno; muy estable y, ampliamente, documentado.
   - Uso típico: Ideal para administración de sistemas y *scripting* estándar.
2. **Zsh (*Z Shell*):**
   - Es una versión más avanzada y configurable que Bash.
   - Ventajas: ofrece autocompletado inteligente, sugerencias en tiempo real y temas visuales (como Oh My Zsh); permite una mayor personalización del *prompt* y funciones avanzadas de historial.

- Desventajas: puede consumir más recursos; requiere configuración inicial para aprovechar sus ventajas.
- Uso típico: Preferido por usuarios avanzados y desarrolladores.

3. **Fish (*Friendly Interactive Shell*):**
   - Diseñado para ser fácil de usar y, visualmente, más amigable.
   - Ventajas: autocompletado automático sin configuración adicional; coloreado de sintaxis por defecto; no requiere editar archivos de configuración complejos.
   - Desventajas: no es totalmente compatible con *scripts* de Bash, lo que limita su uso en entornos de producción.
   - Uso típico: Ideal para usuarios nuevos o tareas interactivas.

En resumen:

- Bash: clásico, estable y ampliamente usado.
- Zsh: más potente y configurable.
- Fish: moderno y simple para principiantes.

(d) *¿Dónde se ubican (path) los comandos propios y externos al Shell?*

En GNU/Linux, los comandos que se ejecutan en el *shell* pueden ser de dos tipos: internos (propios del *shell*) o externos (programas ejecutables del sistema). Cada tipo se encuentra en lugares distintos del sistema de archivos.

1. **Comandos internos (propios del *shell*):**
   - Son instrucciones integradas dentro del propio intérprete (por ejemplo, *cd*, *echo*, *pwd*, *export*, *history*).
   - No son archivos ejecutables en el sistema, sino funciones incorporadas en el shell.
   - Ubicación:
     - Están dentro del intérprete de comandos (por ejemplo, dentro de */bin/bash* o */usr/bin/zsh*).
     - No tienen una ruta propia en el sistema de archivos.
2. **Comandos externos:**
   - Son programas ejecutables almacenados en el sistema de archivos.
   - Cuando el usuario los invoca, el *shell* los busca en los directorios definidos en la variable de entorno $PATH.
   - Ubicaciones comunes:
     - */bin* ⟶ comandos básicos (por ejemplo, *ls*, *cp*, *mv*, *cat*).
     - */usr/bin* ⟶ aplicaciones de usuario (por ejemplo, *grep*, *nano*, *tar*).
     - */sbin* ⟶ comandos administrativos del sistema (por ejemplo, *reboot*, *ifconfig*).
     - */usr/sbin* ⟶ herramientas avanzadas para administración del sistema.

En resumen, los comandos internos están incorporados en el *shell*, mientras que los externos son programas independientes ubicados en los directorios del PATH del sistema.

(e) *¿Por qué se considera que el Shell no es parte del Kernel de GNU/Linux?*

Se considera que el Shell no es parte del Kernel de GNU/Linux porque:

- No gestiona hardware ni recursos del sistema.
- Opera en modo usuario, no en modo núcleo.
- Puede ser reemplazado sin alterar el sistema operativo.
- Actúa como interfaz entre el usuario y el Kernel, no como parte de él.

(f) *¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde se define? ¿Cualquier usuario puede realizar dicha tarea?*

Sí, es posible definir un intérprete de comando distinto para cada usuario. En GNU/Linux, cada usuario puede tener asignado un *shell* diferente como su intérprete de comandos predeterminado. Esto significa que un usuario puede usar, por ejemplo, Bash, otro Zsh y otro Fish, según sus preferencias.

El *shell* predeterminado de cada usuario se define en el archivo */etc/passwd*.

Cada usuario puede cambiar su propio *shell*, siempre que:

- El nuevo intérprete esté instalado en el sistema.
- Esté listado en el archivo */etc/shells*, que contiene los *shells* válidos.

Sólo el usuario *root* (administrador) puede:

- Cambiar el *shell* de otros usuarios.
- Agregar nuevos intérpretes de comandos al sistema.

Ejercicio 6: El Sistema de Archivos (*File System*) en Linux.

(a) *¿Qué es?*

(b) *¿Cuál es la estructura básica de los File System en GNU/Linux? Mencionar los directorios más importantes e indicar qué tipo de información se encuentra en ellos. ¿A qué hace referencia la sigla FHS?*

(c) *Mencionar sistemas de archivos soportados por GNU/Linux.*

(d) *¿Es posible visualizar particiones del tipo FAT y NTFS (que son de Windows) en GNU/Linux?*

Ejercicio 7: Particiones.

Ejercicio 8: Arranque (*Bootstrap*) de un Sistema Operativo.

Ejercicio 9: Archivos y Editores.

Ejercicio 10.

Ejercicio 11.

Ejercicio 12: Procesos.

Ejercicio 13: Proceso de Arranque SystemV (https://github.com/systeminit/si/).

Ejercicio 14: SystemD (https://github.com/systemd/systemd/).

Ejercicio 15: Usuarios.

Ejercicio 16: *File System* y Permisos.

Ejercicio 17: Procesos.

Ejercicio 18: Otros Comandos de Linux (Indicar Funcionalidad y Parámetros).

Ejercicio 19.

Ejercicio 20.

Ejercicio 21.

Ejercicio 22.

Ejercicio 23.

Ejercicio 24.

Ejercicio 25.