



Clase 1

☰ Materia	ISO Práctica
⚙ Estado	Done
📅 Fecha	@August 26, 2022

CONCEPTOS

GENERALES

¿Qué es un sistema operativo?

- Es la parte esencial de cualquier sistema de cómputo.
- Es un programa que actúa como intermediario entre el usuario y el hardware.
- Su propósito es **crear un entorno cómodo y eficiente para la ejecución de programas**
- Su obligación es **garantizar el correcto funcionamiento del sistema**
- Sus funciones principales son: administrar la memoria, la CPU y los dispositivos.

GNU/Linux y software libre

Linux es un sistema operativo tipo **Unix**, pero libre, de **código abierto** lo que nos permite estudiarlo, personalizarlo, auditarlo y aprovecharnos de la documentación.

Está diseñado por miles de programadores, es gratuito y de libre distribución. Existen diversas distribuciones (customizaciones).

GNU = GNU No es Unix.

Iniciado por **Richard Stallman** en **1983** con el fin de crear un *Unix libre*. Para asegurarse de que fuese libre, se necesitó crear un marco regulatorio conocido como **GPL** (General Public License de GNU).

En **1875** Stallman crear la **FSF** (Free software foundation) con el fin de financiar el proyecto GNU.

En **1990** GNU ya contaba con un editor de texto (Emacs), un compiladores (GCC) y gran cantidad de bibliotecas que componen un Unix típico. Faltaba el componente principal: el **núcleo Kernel**.

Se venía trabajando en un núcleo conocido como TRIX, es en **1988** que se decide abandonarlo por su complejidad. En ese momento se adopta como base el núcleo MACH (para crear GNU Hurd que no prosperó). Pero en **1991** *Linux Torvalds* (que venía trabajando desde 1991) crea el núcleo Kernel, el cuál se distribuiría bajo licencia GPL.

En **1992** *Torvalds y Stallman* fusionan ambos proyectos, de ahí nace **GNU/Linux**.

GNU se refiere a 4 libertades principales de los usuarios del software: libertad de usar el programa con cualquier propósito, de estudiar su funcionamiento, de distribuir copias y de mejorar los programas.

Los programas son una forma de expresión de ideas. Son propiedad de la humanidad y deben ser compartidos con todo el mundo.

Características

- Es multiusuario
- Es altamente portable
- Posee diversos interpretes de comandos, de los cuales algunos son programables
- Cada directorio puede estar en una partición diferente (`/temp` , `/home` , etc.)
- Es multitarea y multiprocesador
- Permite el manejo de usuarios y permisos
- Es case sensitive
- Es código abierto
- Todo es un archivo (hasta los dispositivos y directorios)

Diseño

Fue desarrollando buscando la **portabilidad y está diseñado en capas**, dónde hay separación de funciones. Cada capa actúa como una caja negra hacia las otras y posibilidad el desarrollo distribuido.

Tiene soporte para diversos *File Systems*, memoria virtual conformado por la RAM + SWAP. Su desarrollo es mayoritariamente en C y assembler, pero también está formado por otros lenguajes como java, perl, python, etc.

Software libre

- El software libre puede ser usado, copiado, estudiando, modificado y redistribuido libremente, junto a su código fuente.
- Generalmente es de costo nulo (el software libre ≠ software gratuito).
- Se corrigen más rápido las fallas

Software propietario

- Generalmente tiene un costo asociado y no se lo puede distribuir libremente, usualmente no trae su código fuente.
- No permite modificación
- La corrección de fallas está a cargo del propietario, pero tiene menos necesidad de técnicos especializados.

GPL : Generic public license

La licencia pública general de GNU fue creada en el año **1989** por la FSF (Free software foundation). Su propósito es declarar que todo software publicado bajo esta licencia es libre y está protegido teniendo en cuenta las 4 libertades principales ya vistas. La versión actual de la licencia es la 3.

Estructura básica del SO - Núcleo

También conocido como como **Kernel**. Ejecuta programas y gestiona los dispositivos del hardware. Es el encargado de que el software y el hardware puedan trabajar juntos. Su función más importantes es la de administrar la memoria, el CPU y la E/S.

Puede decirse que el kernel **es el sistema operativo**.

Es un núcleo **monolítico híbrido** dónde los drivers y el código del Kernel se ejecutan en modo privilegiado. Su hibridez es por la **capacidad de cargar y descargar funcionalidad a través de módulos**.

Está licenciado bajo la licencia **GPL v2**.

- En 1991 Linus Torvalds inicia la programación de un Kernel Linux basado en Minix (clon de Unix desarrollado por Tenenbaum en 1987 con el fin de crear un S.O. de uso didáctico)
- El 5 de octubre de 1991, se anuncia la primera versión “oficial” de Linux (0.02).
- En 1992 se combina su desarrollo con GNU, formando GNU/Linux.
- La versión 1.0 apareció el 14 de marzo de 1994.
- Desarrollo continuado por miles de programadores al rededor del mundo
- En mayo de 1996 se decide adoptar a Tux como mascota oficial de Linux
- En julio de 1996 se lanza la versión 2.0 y se define la nomenclatura de versionado. Se desarrollo hasta febrero de 2004 y termino con la 2.0.40
- En enero de 1999 se lanza la versión 2.2, que provee mejoras de portabilidad entre otras y se desarrolla hasta febrero de 2004 terminando en la versión 2.2.26
- En 2001 se lanza la versión 2.4 y se deja de desarrollar a fines del 2010 con la 2.4.37.11
 - La versión 2.4 fue la que catapulto a GNU/Linux como un SO estable y robusto. Durante este periodo es que se comienza a utilizar Linux mas asiduamente.
- A fines del 2003 se lanza la versión 2.6
 - Esta versión ha tenido muchas mejoras para el SO dentro de las que se destacan soporte de hilos, mejoras en la planificación y soporte de nuevo hardware
- El 3 de agosto de 2011 se lanza la versión 2.6.39.4 anunciándose la misma desde meses previos como la última en su revisión
- El 17 de julio de 2011 se lanza la versión 3.0
 - No agrega mayores cambios. La decisión del cambio son los 20 años del SO y no superar los 40 números de revisión
 - Totalmente compatible con 2.6
 - La última versión estable es la 4.7.1 (agosto de 2016)

Nomenclatura

- **A:** Denota versión. Cambia con menor frecuencia.
- **B:** Denota mayor revisión. Antes de la versión 2.6, los numeros impares indicaban desarrollo, los pares producción.
- **C:** Denota menor revisión. Solo cambia cuando hay nuevos drivers o características.
- **D:** Cambia cuando se corrige un grave error sin agregar nueva funcionalidad.

Intérprete de comandos

| También conocido como **CLI** (*Command line interface*)

Es el **modo de comunicación entre el usuario y el sistema operativo**. Ejecuta programas a partir del ingreso de comandos. Cada usuario puede tener una interfaz o shell propia, se pueden personalizar y son programables.

Sistema de archivos


Organiza la forma en que se almacenan los archivos en los dispositivos de almacenamiento (fat, nts ext2, ext3, reiser, etc). **Linux adoptó el Extended v2, v3 y v4.**

Desde hace tiempo se está debatiendo el reemplazo de ext por Btrfs (B-tree FS) de Oracle, ya que:

- Soporta mayor tamaño de archivos
- Es más tolerante a fallas y comprobación sin necesidad de desmontar el FS.
- Indexación
- Snapshots
- Compresión
- Defragmentación.

Los directorios más importantes

Los directorios más importantes según **FHS** (Filesystem Hierarchy Standard) son:

-  Tope de la estructura de directorios. Es como el C:\

- `/home` Se almacenan archivos de usuarios (Mis documentos)
- `/var` información que varía de tamaño
- `/etc` Archivos de configuración
- `/bin` Archivos binarios y ejecutables
- `/dev` Enlace a dispositivos
- `/usr` Aplicaciones de usuarios

Distribuciones

Una distribución es una customización de Linux formada por una versión de kernel y determinados programas con sus configuraciones.

MBR

Es un pequeño código que permite arrancar el **sistema operativo**.

El **MBR** es un sector reservado del disco físico (cilindro 0, cabeza 0, sector 1). En todos los disco existe un MBR y, si existiera más de un disco rígido en la máquina, sólo uno es designado como *Primary Master Disk*. El creado con algún utilitario.

El **tamaño del MBR coincide con el tamaño estándar de sector, es decir, 512 bytes**.

Los primeros bytes corresponden al **Master Boot Code** (MBC)

A partir del 446 está la tabla de particiones que es de 64 bytes

Al final existe 2 bytes libres o para firmar el MBR.

La última acción del **BIOS** es leer el MBC, llevarlo a memoria y ejecutarlo. Se se tiene un sistema instalado el bootloader del MBC es típico, en caso de que se tenga más de uno sucede una carga en multietapas.

Particiones

Es una forma de dividir lógicamente el disco físico. Cada sistema operativo es instalado en una partición separada, y cada partición está formateada con un tipo distinto de filesystem.

Es una buena práctica separar los datos de usuario de las aplicaciones y de los sistemas operativos instalados. Y también **ubicar al kernel en una partición de sólo lectura o una que ni siquiera se monta, así no está disponible para los usuarios.**

Aún así particionar tiene desventajas. Debido al tamaño acotado en el **MBR** para la tabla de particiones, se restringe a 4 la cantidad de particiones primarias: 3 primarias y una extendida, con sus respectivas particiones lógicas.



- Una de las 4 particiones puede ser extendida, la cuál se subdivide en volúmenes lógicos.

La **partición primaria**: división cruda del disco (puede haber 4 por disco). Se almacena información de la misma en el MBR.

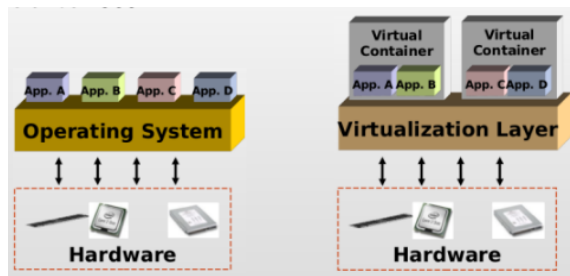
La **partición extendida** sirve para contener unidades lógicas en su interior. Sólo puede existir una partición de este tipo por disco: no se define un tipo de FS directamente sobre ella.

La **partición lógica** ocupa la totalidad o parte de la partición extendida y se le define un tipo de FS. Las particiones de este tipo se conectan como una *lista enlazada*.

Particiones necesarias:

- Como mínimo es necesario una partición para el 
- Es recomendable crear al menos 2 →  y otra para el SWAP.
- Para crearlas se utiliza software denominado **particionador** del cuál existen dos tipos:
 - **Destructivos** → permiten crear y eliminar particiones (*fdisk*)
 - **No destructivos** → permiten crear, eliminar y modificar particiones.

Emuladores



Los emuladores permiten que un equipo pueda correr varios sistemas operativos en forma simultánea compartiendo recursos de hardware. El pionero en esta tecnología fue **IBM** con el **IBM System/360** en los años 1960.

Básicamente se pueden considerar 3 tipos de emuladores:

- **Emulación**

Emulan hardware, tienen que implementar todas las instrucciones de la CPU. Es muy costosa y poco eficiente. Permite ejecutar arquitecturas diferentes a las soportadas por el propio hardware.

- **Virtualización completa**

Permite ejecutar sistemas operativos huéspedes en un sistema anfitrión. Utilizan en el medio un *hypervisor* o monitor de máquinas virtuales. El sistema operativo huésped debe estar soportado en la arquitectura anfitriona. Es más eficiente que la emulación.

- **Paravirtualización**

Permite correr sistemas operativos modificados exclusivamente para actuar en entornos virtualizados. Mayor eficiencia que la virtualización.

La principal diferencia entre **emulador y virtualizador**, es que el virtualizador aprovecha la CPU sobre la que está trabajando y lo hace más veloz; pero el emulador corre cualquier arquitectura, el virtualizado sólo se puede correr la arquitectura virtualizada.

Gestor de arranque

La finalidad del **bootloader** es la de cargar una imagen de Kernel (SO) de alguna partición para su ejecución. Se ejecuta luego del código del **BIOS**.

Existen dos modos de instalación:

- En el **MBR**
- En el sector de arranque de la partición raíz o activa.

Los gestores de arranque pueden ser **GRUB, LILO, NTLDR, GAG, YaST**.

GRUB = GRand Unified Bootloader

Gestor de arranque múltiple más utilizado.

En el MBR sólo se encuentra la fase 1 del que sólo se encarga de cargar la fase 1.5. → La fase 1.5 ubicada en los siguientes 30 KB del disco (MBR gap) carga la fase 2 → La fase 2 es la interfaz de usuario y carga el **kernel seleccionado** → se configura a través del archivo `/boot/grub/menu.lst`

Algunas líneas:

- default: define el SO para defecto a bootear
- timeout: tiempo de espera para cargar el SO por defecto.

En el GRUB 2 la fase 1.5 no existe más, y se añade soporte no ASCII, idiomas y customización. El archivo de configuración ahora es `/boot/grub/grub.cfg` y no debería editarse manualmente, sino que debe usarse el **update-grub**.

El proceso de arranque

Es el proceso de inicio de una máquina y carga del SO, denominado *bootstrap*.

En las arquitecturas x86, el **BIOS** (Basic I/O System) es el responsable de iniciar la carga del SO mediante el **MBC**.

- Esta grabado en un chip (ROM, NVRAM)
- En otras arquitecturas también existe, pero se lo conoce con otros nombres (Power on reset + IPL en *mainframe*, OBP en *SPARC*)

Carga el programa de booteo desde el MBR, el gestor de arranque carga el Kernel (hace disponible los dispositivos, pasa el control al proceso `init`).

BIOS y EFI

El **BIOS** tiene un par de problemas: es de la década del 80, la última acción del mismo es leer el **MBC** del **MBR**, su firmware no facilita la lectura de filesystem y el MBC no puede ocupar más de 446 bytes.

Grub al usar más espacios, usa los 446 del MBR y luego sectores del disco adyacentes que debería estar libres, también conocido como MBR gap.

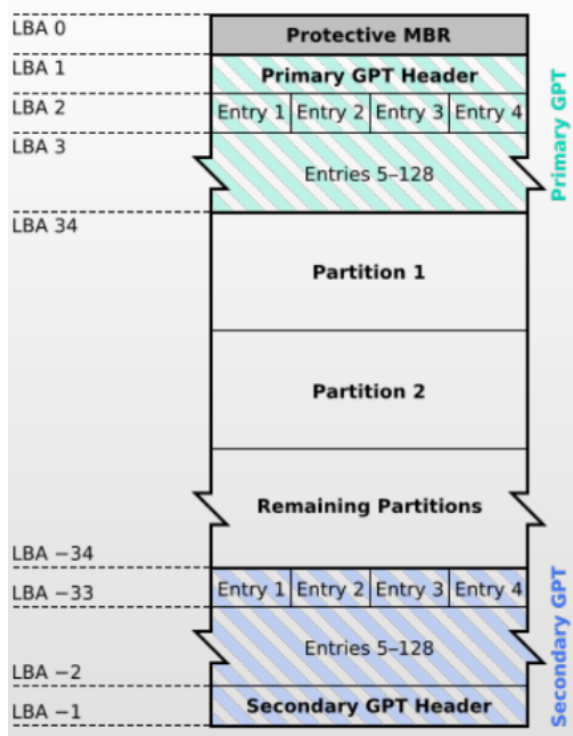
El **EFI** es un nexo entre el SO y el firmware. Utiliza el sistema **GPT** (GUID partition table) para solucionar limitaciones del MBR, como la cantidad de particiones. **GPT**

especifica la ubicación y formato de la tabla de particiones en un disco duro. Es parte de EGI, puede verse como una sustitución del MBR.

EFI es propiedad de intel.

GPT

GUID Partition Table Scheme



Se mantiene un **MBR** para tener compatibilidad con el esquema de **BIOS**.

El modo de direccionamiento de GPT es lógico (logical block addressing) en lugar de *cylinder-header-sector*.

En el LBA 1 está la cabecera GPT. La tabla de particiones en sí está en los bloques sucesivos. La cabecera GPT y la tabla de particiones están escritas al principio y al final del disco (redundancia)

UEFI — Unified Extensible Firmware Interface

Es una alianza entre varias compañías con el objetivo de modernizar el proceso de arranque. UEFI aporta criptografía, autenticación de red y una interface gráfica.

Define la ubicación de gestor de arranque, la interfaz entre el gestor y el firmware. Expone información para los gestores con:

- Información de hardware y configuración del firmware
- Punteros a rutinas que implementan los servicios que el firmware ofrece a los bootloaders u otras aplicaciones UEFI

- Provee un BootManager para cargar aplicaciones UEFI (e.j.: Grub) y drivers desde un UEFI filesystem
- El bootloader ahora es un tipo de aplicación UEFI:
- El Grub será una aplicación UEFI, que reside en el UEFI filesystem donde están los drivers necesarios para arrancar el sistema operativo (FAT32)
- Para el Grub deja de ser necesario el arranque en varias etapas.

Secure Boot

- Propone mecanismos para un arranque libre de código malicioso
- Las aplicaciones y drivers UEFI (imágenes UEFI) son validadas para verificar que no fueron alteradas
- Se utilizan pares de claves asimétricas
- Se almacenan en el firmware una serie de claves publicas que sirven para validar que las imágenes están firmadas por un proveedor autorizado
- Si la clave privada está vencida o fue revocada la verificación puede fallar