Presentación

Introducción a los Sistemas Operativos/Conceptos de sistemas operativos

Facultad de Informática
Universidad Nacional de La Plata

2022

La Cátedra

- Sitio Web: http://catedras.info.unlp.edu.ar (único medio de comunicación)
- Bandas Horarias - Horarios:
  - Turno mañana: Jueves de 8:00 a 9:30 - Aula 5
  - Turno mañana: Viernes de 11:30 a 14:30 - Aula 5
  - Turno tarde: Jueves de 19:00 a 20:30 - Aula 5
  - Turno tarde: Viernes de 17:30 a 20:30 - Aula 4
- Integrantes:
  - JTP turno mañana:
    - Martín Baez: mbaez@info.unlp.edu.ar
  - JTP turno tarde:
    - Emanuel Nucilli: emanuel.nucilli@gmail.com

· Nahuel Cuesta Luengo · Carlos Meza · Alejandro Sebastián
· María Emilia Corrons · Leonardo Otondo · Claudia Quintana
· Adán Mauri · Erica Padovani · Agustín Parmisano
· Carlos Piazza Orlando

Los Parciales

- Entran todos los temas vistos en la práctica (enunciados de TP, explicaciones, material adicional brindado por la cátedra, respuestas al foro)
- La materia consta de 2 parciales:
  - Sistemas Operativos Aplicados (prácticas 1,2 y 3):
    - Primera fecha → 13 de octubre
    - Segunda fecha → 3 de Noviembre
  - Práctica Normal (prácticas 4,5 y 6):
    - Primer fecha → 3 de diciembre
    - Segunda fecha → 22 de diciembre
  - Recuperatorio general: Febrero de 2023
- Para rendir cada parcial se deben contestar las autoevaluaciones de las prácticas correspondientes (**¡OBLIGATORIAS!**)
- Para aprobar la cursada hay que aprobar ambos parciales

Conceptos Generales

Explicación de práctica 1

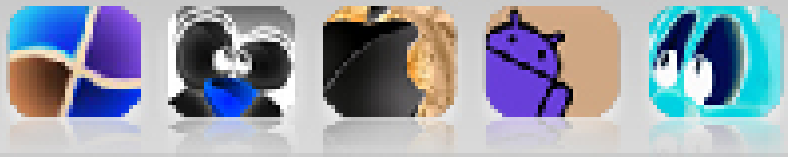
Introducción a los Sistemas Operativos/Conceptos de sistemas operativos

Facultad de Informática
Universidad Nacional de La Plata

2022

¿Qué es un Sistema Operativo?

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



¿Qué es un Sistema Operativo? (cont.)

- Según Wikipedia:
  "...Es un conjunto de programas de computación destinados a realizar muchas tareas..."
- Según un usuario stándar: "Lo que aparece cuando prendo la PC"
- ...

- Es un Sistema Operativo tipo *Unix* (Unix like), pero libre
- S.O. diseñado por miles de programadores
- S.O. gratuito y de libre distribución (se baja desde la Web, CD, etc.)
- Existen diversas distribuciones (customizaciones)
- Es código abierto, lo que nos permite estudiarlo, personalizarlo, auditarlo, aprovecharnos de la documentación, etc...

Podemos ver como está hecho!!!

¿GNU?

- GNU = **G**NU **N**o es **U**nix

- Iniciado por *Richard Stallman* en 1983 con el fin de crear un Unix libre (el sistema GNU)
- Para asegurar que el mismo fuera libre, se necesitó crear un marco regulatorio conocido como GPL (General Public License de GNU)
- En 1985, Stallman crea la FSF (Free Software Foundation), con el fin de financiar el proyecto GNU
- En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y gran cantidad de bibliotecas que componen un Unix típico.
- Faltaba el componente principal → El Núcleo (*Kernel*)

GNU

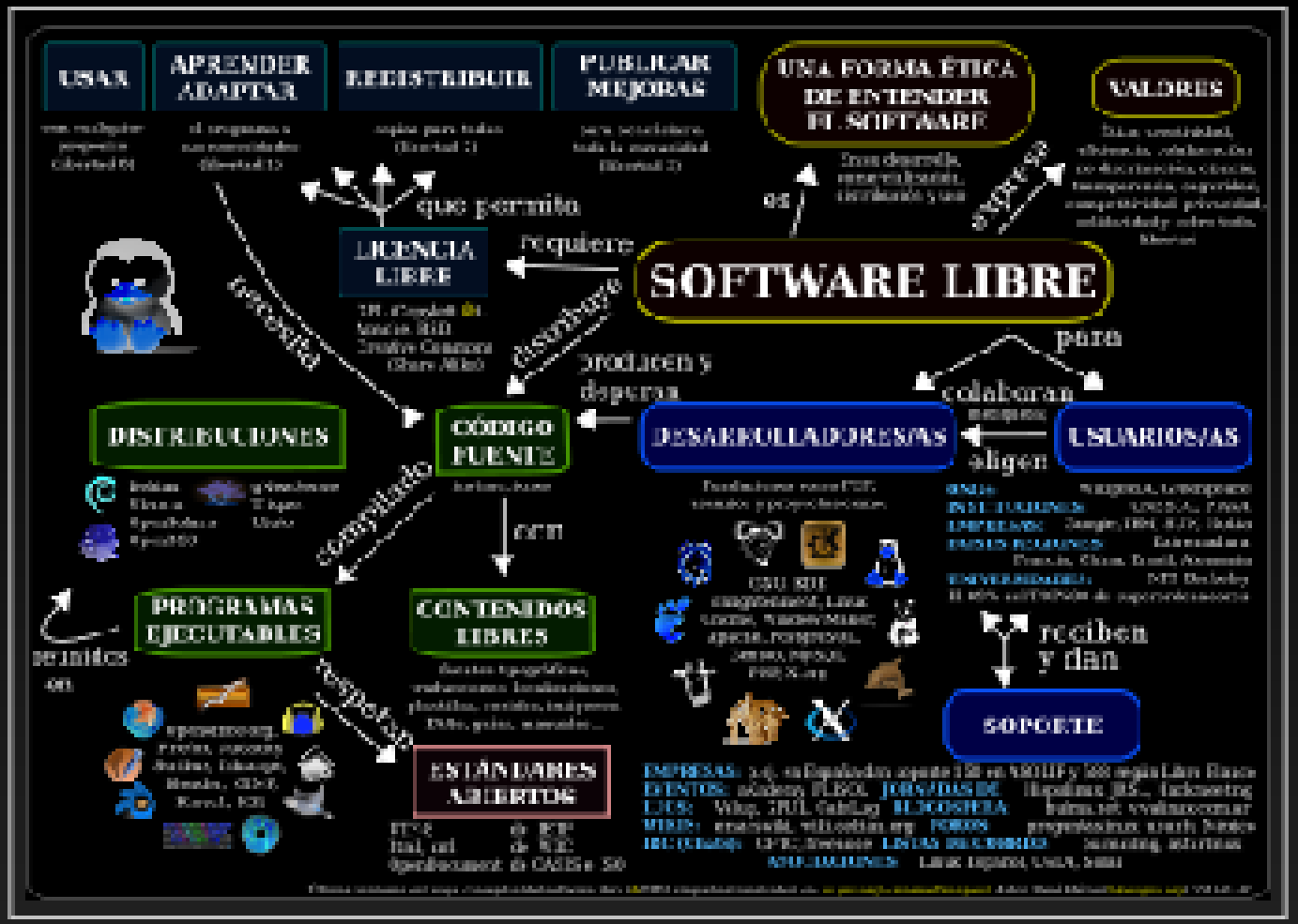
- Si bien ya se venía trabajando en un núcleo conocido como *TRIX*, es en 1988 que se decide abandonarlo debido a su complejidad (corría en hardware muy costoso)
- En este momento se decide adoptar como base el núcleo *MACH* para crear *GNU Hurd*, el cual tampoco prosperó
- *Linus Torvalds* ya venía trabajando desde 1991 en un Kernel denominado *Linux*, el cual se distribuiría bajo licencia GPL
- En el año 1992, Torvalds y Stallman deciden fusionar ambos proyectos, y es allí donde nace *GNU/Linux*
- GNU/Linux pertenece al desarrollo del software libre

GNU (cont.)

- GNU se refiere a 4 libertades principales de los usuarios del software:
  - Libertad de usar el programa con cualquier propósito
  - Libertad de estudiar su funcionamiento
  - Libertad para distribuir sus copias
  - Libertad para 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"

¿Software libre?



- Características del software libre:
  - Una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente
  - Generalmente es de costo nulo ← Es un gran error asociar el software libre con el software gratuito ← Pensar en software gratis que se distribuye con restricciones
  - Es común que se distribuya junto con su código fuente
  - Corrección más rápida ante fallas
  - Características que se refieren a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software

¿Software propietario?

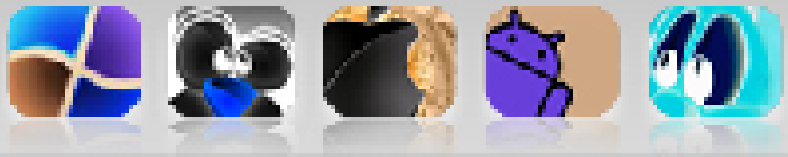
- Características del software propietario:
  - Generalmente tiene un costo asociado
  - No se lo puede distribuir libremente
  - Generalmente no permite su modificación
  - Normalmente no se distribuye junto con su código fuente
  - La corrección de fallas esta a cargo del propietario
  - Menos necesidad de técnicos especializados

GPL: Generic Public License

- Licencia Pública General de GNU
- Creada en el año 1989 por la FSF
- Su objetivo principal es proteger la libre distribución, modificación y uso del software GNU
- Su propósito es declarar que todo software publicado bajo esta licencia, es libre y esta protegido teniendo en cuenta las 4 libertades principales ya vistas
- La versión actual de la licencia es la 3

Características generales de GNU/Linux

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



Diseño

- Fue desarrollado buscando la portabilidad de los fuentes
- Desarrollo en capas
  - Separación de funciones
  - Cada capa actúa como una caja negra hacia las otras
  - Posibilita el desarrollo distribuido
- Soporte para diversos File Systems
- Memoria virtual = RAM + *SWAP*
- Desarrollo mayoritario en C y assembler
- Otros lenguajes: java, perl, python, etc.

Estructura básica del S.O. - Núcleo

- También conocido como *Kernel*
- Ejecuta programas y gestiona dispositivos de hardware
- Es el encargado de que el software y el hardware puedan trabajar juntos
- Sus funciones más importantes son la administración de memoria, CPU y la E/S
- En si, y en un sentido estricto, es el sistema operativo
- Es un núcleo monolítico híbrido:
  - Los drivers y código del Kernel se ejecutan en modo privilegiado
  - Lo que lo hace híbrido es la capacidad de cargar y descargar funcionalidad a través de módulos
- Está licenciado bajo la lecencia GPL v2

Núcleo - Un poco de historia

- En 1991 Linus Torvalds inicia la programacion de un Kernel *Linux* basado en *Minix* (clon de Unix desarrollado por *Tenembaum* 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

Núcleo - Un poco de historia (cont.)

- 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 desarrolló hasta febrero de 2004 y terminó con la 2.0.40
- En enero de 1999 se lanza la versión 2.2, que proveé 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 catapultó a GNU/Linux como un SO estable y robusto. Durante este período es que se comienza a utilizar Linux más asiduamente

Núcleo - Un poco de historia (cont.)

- 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[1]
  - 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)

[1] http://kernelnewbies.org/Linux_3.0

- **A**: Denota versión. Cambia con menor frecuencia. En 1994 (versión 1.0) y en 1996 (versión 2.0)
- **B**: Denota mayor revisión. Antes de la versión 2.6, los números 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 ← Casi no se usa en las ramas 3.x y 4.x, viendose reflejado en **C**

Iterprete de comandos

- También conocido como CLI (Command Line Interface)
- Modo de comunicación entre el usuario y el SO
- Ejecuta programas a partir del ingreso de comandos
- Cada usuario puede tener una interfaz o shell
- Se pueden personalizar
- Son programables
- Bourne Shell (sh), Korn Shell (ksh), Bourne Again Shell (bash)(autocompletado, history, alias)

Sistemas de archivos

- Organiza la forma en que se almacenan los archivos en dispositivos de almacenamiento (fat, ntfs ext2, ext3, reiser, etc.)
- El adoptado por GNU/Linux es el Extended (v2, v3, v4)
- Hace un tiempo se está debatiendo el reemplazo de ext por Btrfs (B-tree FS) de Oracle
  - Soporte de mayor tamaño de archivos
  - Más tolerante a fallas y comprobación sin necesidad de desmontar el FS
  - Idexación
  - Snapshots
  - Compresión
  - Defragmentación

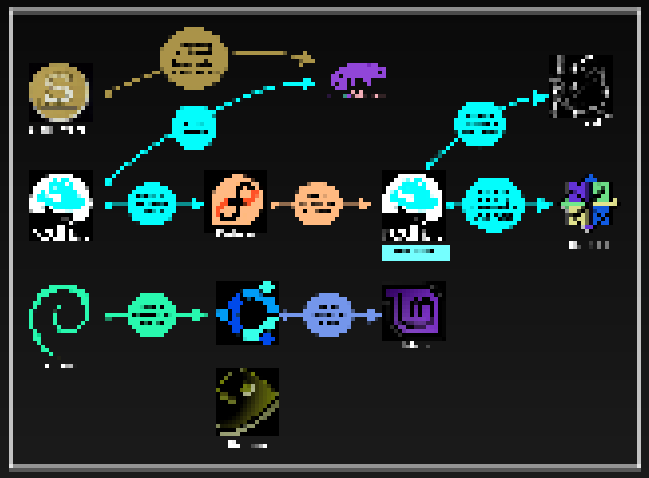
Sistemas de archivos - Directorios más importantes

- Directorios más importantes según FHS (Filesystem Hierarchy Standard)
  - / Tope de la estructura de directorios. Es como el C:\
  - /home Se almacenan archivos de usuarios (Mis documentos)
  - /var Información que varía de tamaño (logs, BD, spools)
  - /etc Archivos de configuración
  - /bin Archivos binarios y ejecutables
  - /dev Enlace a dispositivos
  - /usr Aplicaciones de usuarios

Estructura básica del S.O. - Utilidades

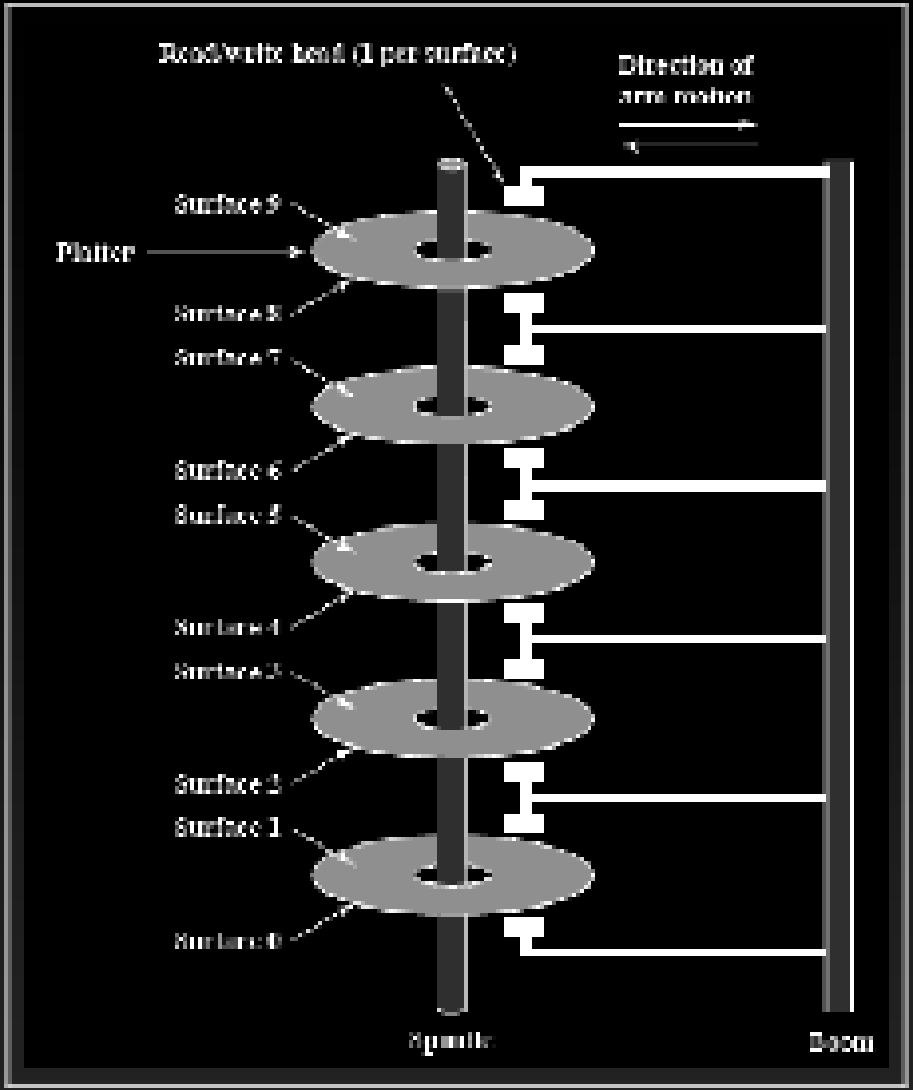
- Paquete de software que permite diferenciar una distribución de otra.
- Editores de texto:
  - vi
  - emacs
  - joe
- Herramientas de networking:
  - wireshark
  - tcpdump
- Paquetes de oficina:
  - OpenOffice
- Iterface gráficas:
  - GNOME / CINNAMON
  - KDE
  - LXDE

- Una distribución es una customización de GNU/Linux formada por una versión de kernel y determinados programas con sus configuraciones[2]

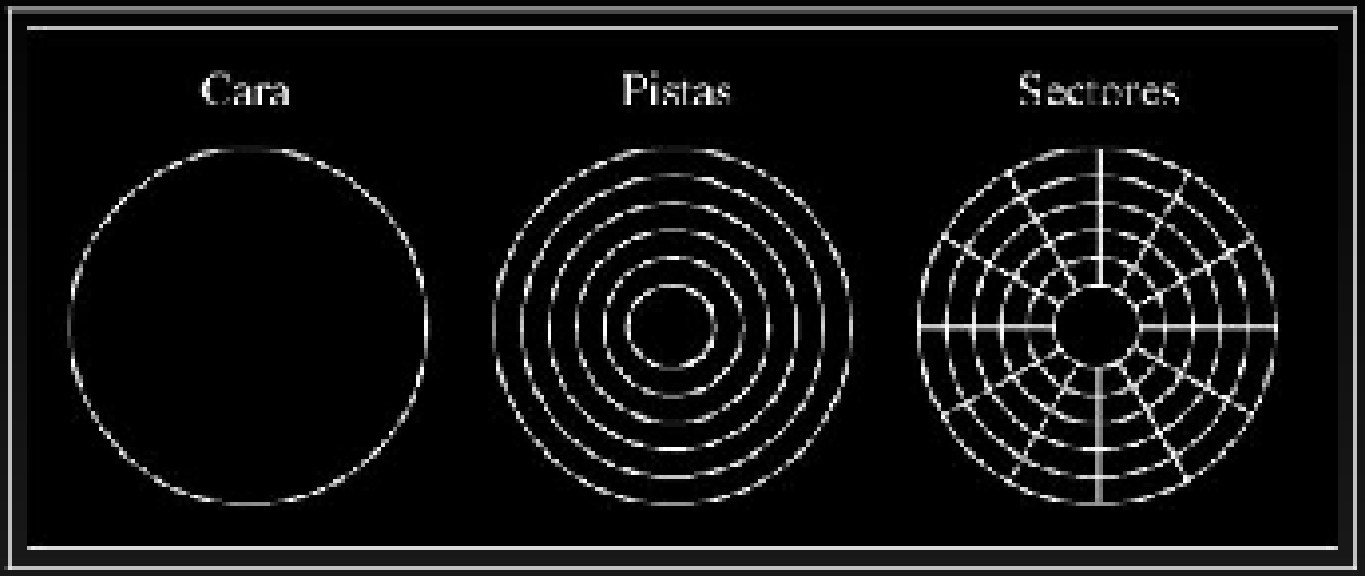


[2] http://www.linux.com/directory/Distributions

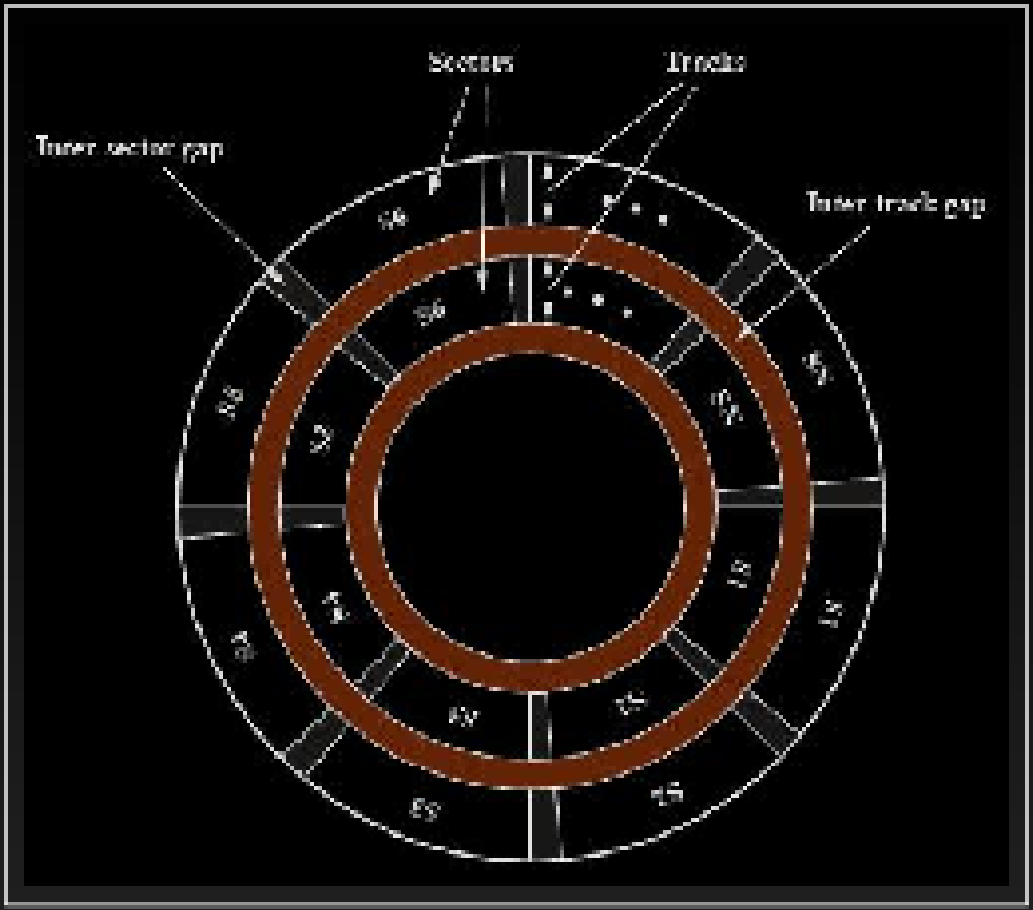
Organización física de discos



Organización física de discos (cont.)



Organización física de discos (cont.)



Conceptos para la instalación - **MBR**

- Sector reservado del disco físico (cilindro 0, cabeza 0, sector 1)
- Existe un MBR en todos los discos
- Si existiese más de un disco rígido en la máquina, sólo un es designado como *Primary Master Disk*

Conceptos para la instalación - MBR (cont.)

- El tamaño del MBR coincide con el tamaño estándar de sector, 512 bytes:
  - Los primeros bytes corresponden al *Master Boot Code* (MBC)
  - A partir del byte 446 está la tabla de particiones. Es de 64 bytes
  - Al final existen 2 bytes libres o para firmar el MBR
- Es creado con algún utilitario
- El MBC es un pequeño código que permite arrancar el SO
- La última acción del *BIOS* es leer el MBC. Lo lleva a memoria y lo ejecuta
- Si se tiene un sistema instalado → bootloader de MBC típico. Sino → uno diferente (*multietapa*)

Conceptos para la instalación - Particiones

- Es una forma de dividir lógicamente el disco físico:
  - *DOS* y *W95* no pueden manejar filesystems mayores a 2GB
  - Cada sistema operativo es instalado en una partición separada
  - Cada partición se formatea con un tipo de filesystem destino (*fat*, *ntfs*, *ext*, etc.)
  - Es una buena práctica separar los datos del usuario de la aplicaciones y/o sistema operativo instalado
  - Tener una partición de *restore* de todo es sistema
  - Poder ubicar el Kernel en una partición de solo lectura, o una que ni siquiera se monta (no está disponible para los usuarios)
  - Particionar demasiado un disco puede tener desventajas: ¡pensar..!

Conceptos para la instalación - Particiones (cont.)

- 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 cual se subdivide en volúmenes lógicos
- Partición primaria: división cruda del disco (puede haber 4 por disco). Se almacena información de la misma en el MBR
- Partición extendida: sirve para contener unidades lógicas en su interior. Solo puede existir una partición de este tipo por disco. No se define un tipo de FS directamente sobre ella
- 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

Conceptos para la instalación - Particiones (cont.)

- Como mínimo es necesario una partición (para el /)
- Es recomendable crear al menos 2 (/ y *SWAP*)
- Para crearlas, se utiliza software denominado **particionador**. Existen 2 tipos:
  - Destructivos: permiten crear y eliminar particiones (fdisk)
  - No destructivo: permiten crear, eliminar y modificar particiones (fips, gparted) ← generalmente las distribuciones permiten hacerlo desde la interfaz de instalación
- ¿Para qué podríamos crear otras particiones?

Conceptos para la instalación - Particiones (cont.)

- Como mínimo es necesario una partición (para el /)
- Es recomendable crear al menos 2 (/ y *SWAP*)
- Para crearlas, se utiliza software denominado **particionador**. Existen 2 tipos:
  - Destructivos: permiten crear y eliminar particiones (fdisk)
  - No destructivo: permiten crear, eliminar y modificar particiones (fips, gparted) ← generalmente las distribuciones permiten hacerlo desde la interfaz de instalación
- ¿Para qué podríamos crear otras particiones?

Nuestro ambiente de trabajo

- Vamos a trabajar en un ambiente controlado → *VirtualBox*
- Necesitamos crear una máquina virtual y asignarle recursos
- Booteamos la máquina virtual iniciando desde algún medio de instalación
- Seguimos las instrucciones de instalación
- Verificamos que podamos arrancar el/los sistemas operativos instalados

Particionado de discos

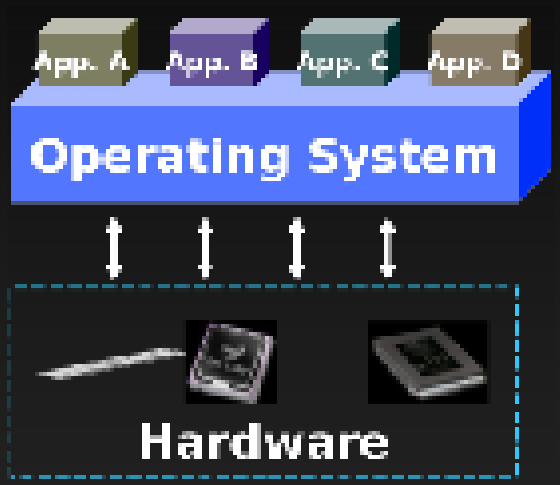
- Particionando - 3 escenatios posibles:
  - Usar espacio libre no particionado
  - Usar partición no usada
  - Usar espacio libre de una partición actvia (más complicado):
    - Cambio destructivo
    - Cambio no destructivo
- En nuestra instalación:
  - /dev/hda1: DOS con Wndows (2 GB) ← ¡es mucho!
  - /dev/hda2: /boot → 60 MB aproximadamente
  - /dev/hda3: / → 6 GB aproximadamente
  - /dev/hda4: área de intercambio (SWAP)

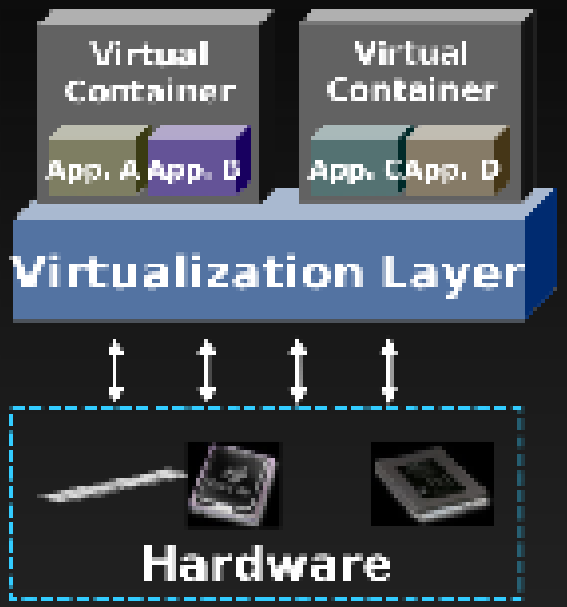
Verificando la configuración de nuestro disco

- Desde Windows: utilizamos el administrador de discos
- Desde DOS: usamos fdisk
- Para instalar un nuevo SO necesitamos espacio libre sin particionar
- Si no lo tenemos, debemos generarlo. Para esto existen diversos escenarios:
  - Eliminar una partición existente
  - Redimensionar una partición
- ¿Qué ocurre en cada caso? ¿Qué software vamos a usar?

Emuladores/Virtualizadores

- Para los fines de este curso, permiten virtualizar plataformas
- Permite que en un equipo puedan correr varios SO en forma simultánea compartiendo recursos de hardware
- El pionero en esta tecnología fue *IBM* con el IBM System/360 en los años 1960





Emuladores/Virtualizadores (cont.)

- Básicamente se pueden considerar 3 tipos:
  - 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 hardware
  - Virtualización completa:
    - Permiten ejecutar SO huéspedes en un sistema anfitrión (host)
    - Utilizan en el medio un hypervisor o monitor de máquinas virtuales
    - El SO huésped debe estar soportado en la arquitectura anfitriona
    - Es más eficiente que la emulación (*Intel-VT* y *AMD-V*)
  - Paravirtualización:
    - Permite correr SOs modificados exclusivamente para actuar en entornos virtualizados
    - Mayor eficiencia que la virtualización

Emuladores/Virtualizadores (cont.)

- Las principales diferencias entre ellos son:
  - Los virtualizadores aprovechan el CPU sobre la que están trabajando, lo cual los hace más veloces
  - En un emulador se puede correr cualquier arquitectura. En un virtualizador solo se puede correr la arquitectura virtualizada

Gestor de arranque

- La finalidad del bootloader es la de cargar una imagen de *Kernel* (sistema operativo) de alguna partición para su ejecución
- Se ejecuta luego del código del BIOS
- Existen 2 modos de instalación:
  - En el MBR (puede llegar a utilizar *MBR gap*)
  - En el sector de arranque de la particion raíz o activa (*Volume Boot Record*)
- *GRUB*, *LILO*, *NTLDR*, *GAG*, *YaST*, etc.

Grub Legacy

- **GRand Unified Bootloader**: gestor de arranque múltiple más utilizado
- En el MBR solo se encuentra la fase 1 del que solo se encarga de cargar la fase 1.5
- Fase 1.5: ubicada en los siguientes 30 KB del disco (*MBR gap*). Carga la fase 2
- Fase 2: interfaz de usuario y carga el Kernel seleccionado
- Se configura a través del archivo /boot/grub/menu.lst
- Algunas líneas:
  - default: define el SO por defecto a bootear
  - timeout: tiempo de espera para cargar el SO por defecto

```
title Debian GNU/Linux
root (hd0,1) #(Disco,Particion)
kernel /vmlinuz-2.6.26 ro quiet root=/dev/hda3
initrd /initrd-2.6.26.img
```

- Utilizado en la mayoria de las distribuciones
- Dentro de sus mejoras, incluye el soporte a nuevas arquitecturas, soporte de caracteres no ASCII, idiomas, customización de menúes, etc.
- En Grub 2 la fase 1.5 ya no existe más
- El archivo de configuración ahora es /boot/grub/grub.cfg y no debería editarse manualmente → update-grub
- Más información en: `https://help.ubuntu.com/community/Grub2`

Proceso de arranque

- Es el proceso de inicio de una máquina y carga del sistema operativo y se denomina *bootstrap*
- En las arquitecturas x86, el **BIOS (Basic I/O System)** es el responsable de iniciar la carga del SO a través del MBC:
  - Está grabado en un chip (*ROM*, *NVRAM*)
  - En otras arquitecturas también existe, pero se lo conoce con otro nombre:
    - Power on Reset + IPL en *mainframe*
    - OBP (OpenBoot PROM): en *SPARC*
- Carga el programa de booteo (desde el MBR)
- El gestor de arranque lanzado desde el MBC carga el Kernel:
  - Prueba y hace disponibles los dispositivos
  - Luego pasa el control al proceso **init**
- El proceso de arranque se ve como una serie de pequeños programas de ejecución encadenada

Problemas del arranque basado en BIOS

- Es de la década del 80
- La última acción del BIOS es leer el MBC del MBR
- El firmware del BIOS no facilita la lectura de filesystems
- El MBC no puede ocupar más de 446 bytes
  - Por ejemplo Grub, utiliza los 446 bytes del MBR y luego utiliza sectores del disco adyacentes que deberían estar libres (*MBR gap*)

Extensible Firmware Interface

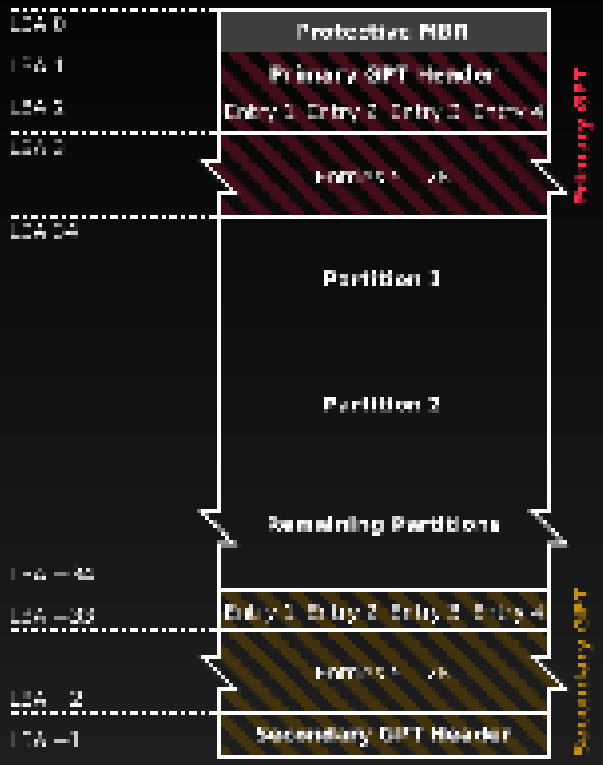
- EFI: nexo entre el SO y el firmware
- Utiliza el sistema **GPT** (GUID partition table) para solucionar limitaciones del MBR, como la cantidad de particiones
- GPT especifica la ubicación y formato de la tabla de particiones en un disco duro
- Es parte de EFI. Puede verse como una sustitución del MBR
- La especificación EFI es propiedad de Intel

GPT

- Se mantiene un MBR para tener compatibilidad con el esquema BIOS
- GPT usa un modo de direccionamiento lógico (logical block addressing **LBA**) en lugar de *cylinder-header-sector*
- El MBR "heredado" se almacena en el LBA 0
- 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)

GPT (cont.)

GUID Partition Table Scheme



- UEFI Forum
  - Alianza entre varias compañias con el objetivo de modernizar el proceso de arranque
  - Representantes de AMD, American Megatrends, Apple, HP, Dell, IBM, Insyde Software, Intel, Lenovo, Microsoft, Phoenix Technologies
  - EFI es propiedad de Intel
  - UEFI es propiedad del UEFI Forum
  - UEFI aporta criptografía, autenticación de red y una interface gráfica

UEFI (cont.)

- Define la ubicación de gestor de arranque
- Define la interfaz entre el gestor de arranque y el firmware
- Expone información para los gestores de arranque 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 booloader 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 estén firmadas por un proveedor autorizado
- Si la clave privada está vencida o fue revocada la verificación puede fallar

¿Preguntas?