# SISTEMAS DE ARCHIVOS

1 de febrero

Campanita

# Antecedentes

- **Todo es una secuencia o 'stream' de bytes**
- Los primeros dispositivos no reconocían archivos, solo almacenaban conjuntos de datos:
  - **Analógicos:** discos de vinilo, cintas de casette
  - En el nivel más bajo, los dispositivos digitales también almacenan solamente bytes

# ¿Que és un archivo?

- **Abstracción o TAD que el SO propone para un conjunto de bytes en un dispositivo.**
- El SO expone funciones al usuario para poder manipular el contenido de un archivo (o sea, los bytes asociados) sin preocuparse de cómo se guardan éstos en el dispositivo.

# ¿Que és un archivo?

- **Abstracción o TAD que el SO propone para un conjunto de bytes en un dispositivo.**
- El SO expone funciones genéricas al usuario para poder manipular el contenido de un archivo (o sea, los bytes asociados) sin preocuparse de cómo se guardan éstos en el dispositivo:

# Operaciones sobre archivos

- El SO expone funciones genéricas al usuario para poder manipular el contenido de un archivo (o sea, los bytes asociados) sin preocuparse de cómo se guardan éstos en el dispositivo:

# Operaciones sobre archivos

- Crear (create)
- Borrar (delete)
- Abrir (open)
- Cerrar (close)
- Leer (read)
- Escribir (write, flush)
- Reposicionar (seek)

# Modos de apertura

Un archivo puede abrirse en distintos modos, lo que permite un conjunto distinto de operaciones sobre el mismo

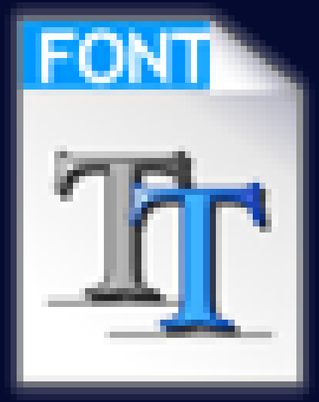
- r: para lectura, si no existe, falla
- w: para escriura, si no existe, falla
- a: para añadido, si no existe, se crea
- r+: para lectura/escritura, si no existe, falla
- w+: para lectura/escritura. si no existe, se crea
- a+: para añadido, si no existe, se crea

# Archivos de texto y binarios

- **Los "archivos de texto" son archivos binarios que solo contienen un subconjunto de bytes, los que representan caracteres imprimibles**
- Tradicionalmente, el conjunto de caracteres imprimibles ASCII
  - Ahora, con la adopción de Unicode, cualquier caracter Unicode válido que no sea un caracter de control
  - Aunque depende del contexto.

# Formato de archivo (I)

- **Los datos dentro de un archivo pueden estar ordenados de mil formas distintas, y corresponde a quien los crea decidir el orden de los datos en un archivo**.
  - El formato de un tipo de archivo específico puede estar reglamentado por un estándar o especificación




# Formato de archivo (II)

- Existen formatos libres y privativos
- Existen formatos cuya especificación es libre y otros cuya especificación no lo es
- Existen formatos documentados, documentados a medias, y no documentados
- Generalmente, los formatos libres están bien documentados

# Formato de archivo (III)

- ¿Cómo determinar el tipo de un archivo?
  - Extensiones de archivo
    - Implementados originalmente por CP/M
      - Y, por consiguiente, MS/DOS y Windows
  - Metadatos asociados a un archivo
    - Mac OS y typecodes
  - Números mágicos
    - Forma seguida por los sistemas Unix
- Hay qué tener cuidado al hacer esto.

# Sistema de archivos (I)

- **Los sistemas de archivos organizan archivos**
- Algunos sistemas de archivos buscan redundancia, mayor rapidez, tolerancia a fallos, protección por cifrado.
- Permiten guardar diversos datos sobre los archivos: "metadatos":
  - Nombre, permisos, fechas (de acceso, de modificación), si son de alguna categoría especial
- **Ruta de archivo:** existente para prácticamente cualquier SO

# Sistema de archivos (II)

- FAT
  - FAT16
  - FAT32
  - exFAT
- NTFS
- ext*
  - ext2
  - ext3
  - ext4
- HFS
  - HFS+
- UFS
- ZFS
- btrfs
- cdfs
  - UDF (ISO 9600)

# Sistema de archivos (III)

- En general, la abstracción del sistema de archivos se parece a un archivero de oficina en la vida real
- La nomenclatura puede variar un poco, dependiendo del enfoque y del tipo de usuario final: directorios en Unix vs carpetas en Windows

# Sistema de archivos virtual (I)

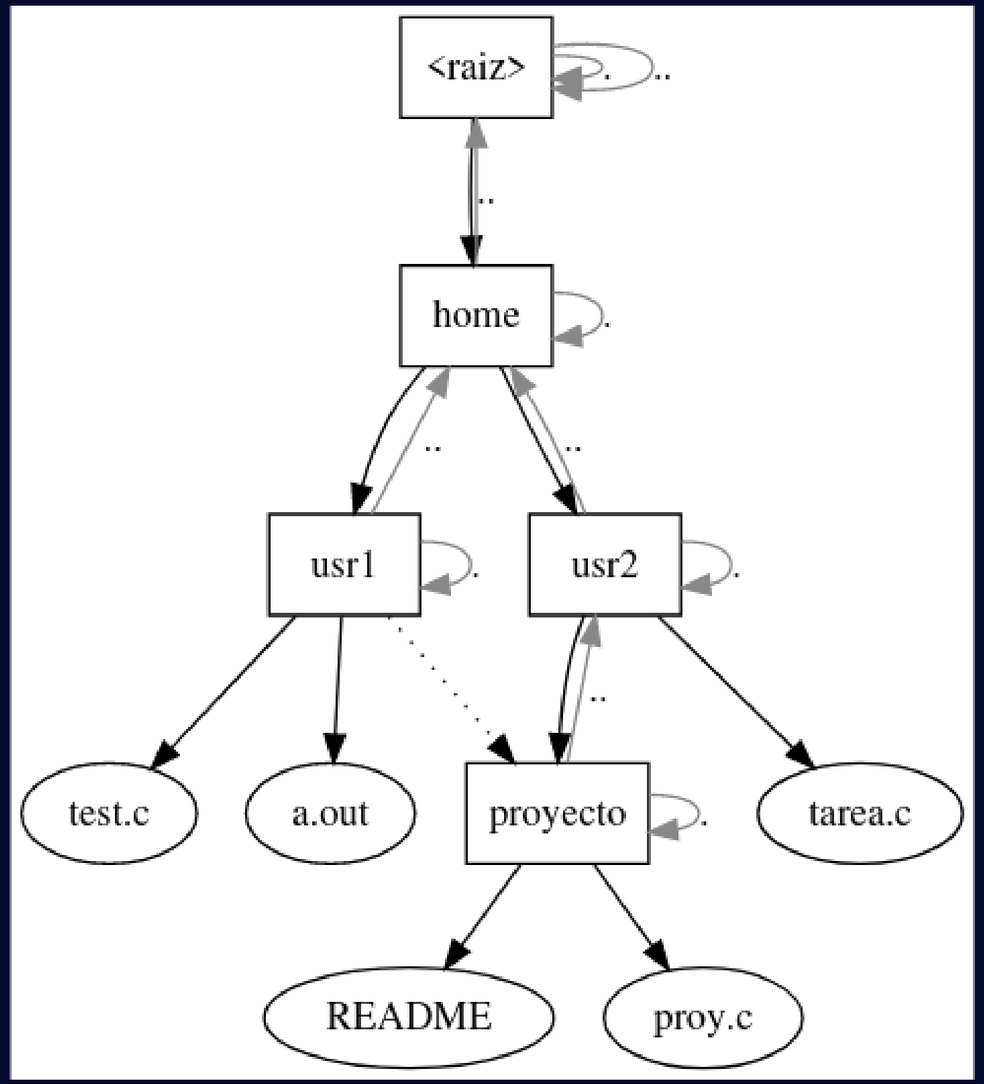
- La idea de sistema de archivos tiene más usos, y puede ser usada para exponer datos que no necesariamente están en un soporte.
- Los sistemas de archivos que funcionaban sobre datos en un soporte se categorizan ahora como 'sistemas de archivos de disco'
  - **Sistemas de archivos de red:** sshfs, nfs, ftp
  - **Sistemas de archivos de propósito especial:** tmpfs, swap, lufs, devfs, sysfs

# Sistema de archivos virtual (II)

- **Ejemplos:**
  - Las rutas de una URL tienen forma de ruta de archivo, pero no necesariamente tienen qué ser archivos, por eso siempre hablamos de *recursos*
  - **Filosofía Unix:** todo es un archivo
  - Archivos especiales: pipas, dispositivos,
  - **FUSE:** crea tu propio sistema de archivos

# Más temas

- Jerarquia como grafos
- Ligaduras
  - Duras
  - Simbólicas
  - Accesos directos
- Descriptores de archivo
- Buffers



# Montar imagen

```
# mount -t ext4 -o ro,loop
/tmp/imagen5.img /mnt/tmp/
```

# PERMISOS UNIX

# Propiedad

- Todos los archivos y directorios en un sistema Unix pertenecen a un usuario y grupo en específico.
- Solo el propietario de un archivo, y el usuario root pueden cambiar estos permisos.
- Esto es parte del estándar POSIX.

```
drwxr-s---   2 Debian-exim       adm               4.0K feb  1 06:52 exim4
-rw-r--r--   1 root              root                 0 ene  9 11:47 faillog
-rw-r--r--   1 root              root               6.1K ene 28 14:39 fontconfig.log
drwxr-xr-x   3 root              root               4.0K ene  9 12:13 installer
drwxr-sr-x+  3 root              systemd-journal    4.0K ene  9 12:14 journal
-rw-rw-r--   1 root              utmp                 0 ene  9 11:47 lastlog
```

# chown, chgrp

Dos comandos nos permitirán modificar el usuario y grupo dueño de un archivo: **chown** y **chgrp**.

*comando* [-RL | -h] *usuario/grupo objetivo*

- Parámetros específicos:
  - -R aplica recursivamente
  - -L atraviesa enlaces simbólicos a directorios
  - -h aplica la acción sobre el enlace simbólico, en vez de sobre el objeto apuntado

# Permisos

Todos los archivos y directorios en un sistema Unix tienen 3 tipos de permisos básicos, que indican qué usuarios pueden operar con ellos, y cómo:

| Permiso | Archivos | Directorios |
|---|---|---|
| **Lectura (r)** | Leer el archivo | Listar archivos dentro del directorio (ls) |
| **Escritura (w)** | Escribir sobre el archivo | Modificar entradas del directorio: crear remombrar y borrar archivos |
| **Ejecución (x)** | Ejecutar el archivo | Moverte al directorio (cd), o buscar dentro del directorio |

# setuid y setgid

Existe forma de otorgar ciertos permisos ‘elevados’ a un usuario común cuando ejecuta un programa.

- **Sobre ejecutables**: otorga al usuario los permisos del dueño del ejecutable (*setuid*), o del grupo (*setgid*).
- **Sobre directorios**: *setuid* es ignorado, *setgid* obliga a que todos los archivos dentro del directorio tengan el mismo grupo dueño.

# sticky

- Dos permisos diferentes, en realidad:
  - **Sobre ejecutables**: hace que un ejecutable se quede residente en memoria principal.
  - **Sobre directorios**: evita que un usuario modifique detalles sobre archivos que no posee.
- El uso sobre ejecutables ya no se implementa en casi ningún sistema Unix
  - Sí en UnixWare y HP-UX

# chmod

El comando para modificar permisos en archivos es chmod.

chmod *permisos objetivo*

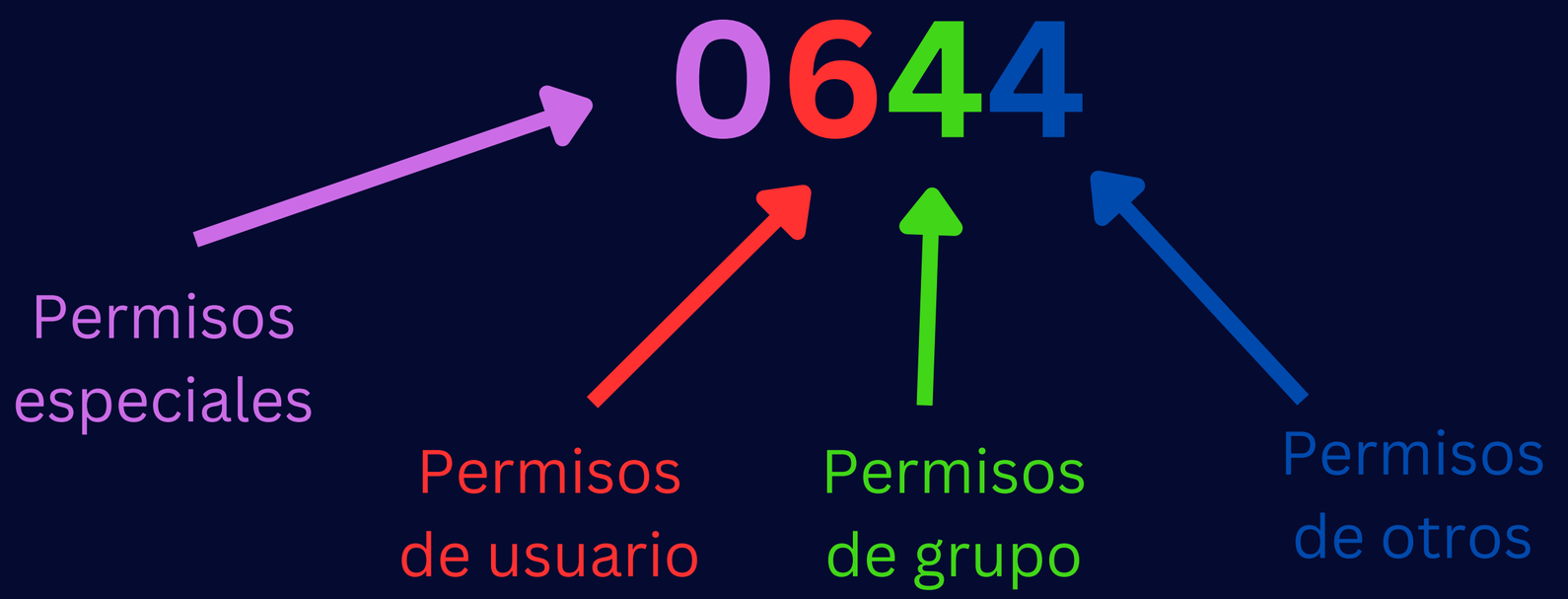
- Parámetros específicos:
  - -R aplica recursivamente
  - -L atraviesa enlaces simbólicos a directorios
- chmod no cambia permisos de enlaces simbólicos, puesto que éstos nunca se ocupan en el sistema

# Notación

Para representar los permisos que un archivo tiene asociados, tenemos dos notaciones:

- **Numérica o absoluta**: un número octal de tres dígitos, un dígito por agente
- **Simbólica**: una cadena de letras, donde cada letra especifica un permiso. Además, incluye otros datos.

# Notación numérica



# Operaciones con chmod (I)

Establecer permisos base con chmod es sencillo, ejemplos:

- chmod 644 archivo.txt

Establecer los bits setuid, setgid y sticky se hace anteponiendo un dígito octal que establece los tres permisos en el orden descrito:

- chmod 1644 archivo.txt

E

# Notación simbólica (I)

# Tipo de inodo

- Regular **(-)**
- Directorio **(d)**
- Enlace simbólico **(l)**
- FIFO **(p)**
- De bloque **(b)**
- De caracter **(c)**
- Socket **(s)**

# Notación simbólica (II)

- En la posición para indicar el permiso ejecutable, pueden haber otros caracteres, que depende de si los bits especiales están activos:
  - Si setuid o setgid están activos: **s** o **S**, dependiendo de si el archivo es o no ejecutable
  - Si sticky está activo, **t** o **T**, dependiendo de si el archivo es o no ejecutable
- Pueden aparecer estos símbolos al final:
  - **+** si existen permisos adicionales
  - **-** si existe un contexto SELinux
  - **@** si existen atributos de archivos extendidos

# Operaciones con chmod (II)

Añadir permisos:

*tipos+permisos*

Quitar permisos:

*tipos-permisos*

Establecer permisos:

*tipos=permisos*

# Ejercicios

- chmod a+r myfile
- chmod +r myfile
- chmod go-rw myfile
- chmod a-w+x myfile
- chmod go=r myfile
- chmod o-w mydir
- chmod o-w
- chmod -R o+x mydir
- chmod u-x,g+wx,o=x myfile

- chmod 751 myfile
- chmod 744 myfile

- -rwxr-xr-x
- crw-rw-r--
- dr-x------
- -rwsr-Sr-t

# Problemas entre sistemas

- No todos los sistemas implementan POSIX:
  - FAT solo tiene un conjunto de permisos para un solo tipo de usuario
  - NTFS (y Windows) implementa un sistema completamente diferente y más complejo con ACLs:
  - HFS y HFS+ no implementan permisos
- Algunos solo implementan un subconjunto básico:
  - Linux implementa experimentalmente permisos NFS
  - macOS también, desde la versión 10.4 Tiger (¡pero recomiendan no usarlo!)
- Otros, lo implementan con variaciones:
  - OpenVMS implementa 4 grupos (sistema, usuario, grupo, otros)

# ¡GRACIAS!

https://mothereff.in/utf-8