Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Bloque Temático 3: Gestión de Archivos
Unidad Temática 8

Implementación de Directorios y Protección

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

Implementación de directorios

- **Objetivos**
  - Estudiar el concepto de **directorio**
  - Entender los conceptos de enlace físico y enlace simbólico
  - Analizar las **técnicas** de gestión del espacio libre de disco
  - Describir el mecanismo de protección de la información utilizado en sistemas UNIX
- **Bibliografía**
  - A. Silberschatz, P.B. Galvin: “Fundamentos de Sistemas Operativos”, McGraw-Hill, 7ª ed. 2006 (Capítulos 10 y 11)

- **Contenido**
  - **Concepto de directorio**
  - Implementación de directorios
  - Enlaces o referencias a archivos
  - Gestión del espacio de disco
  - Protección

- **Arquitectura del sistema de archivos: Visión Usuario**

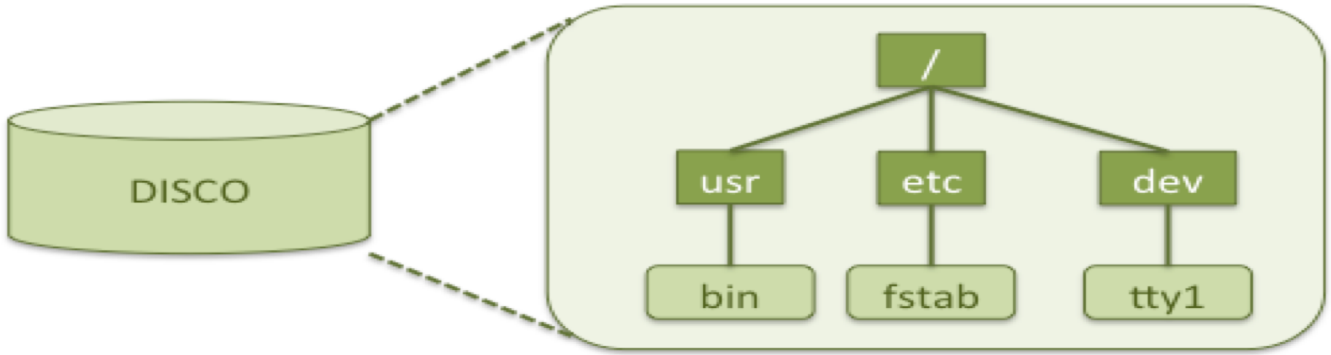
Nivel Usuario
Abstracciones de Archivo y Directorio

Bibliotecas Usuario (para operar con archivos)
Interfaz con las llamadas al sistema sobre archivos y directorios

Operaciones sobre archivos:
- **Abrir y cerrar** archivos
- **Leer/Escribir** sobre archivo
- **Posicionarse** dentro de un archivo

Operaciones sobre directorios:
- **Crear/Borrar entradas** a directorio
- **Renombrar** archivos
- Buscar por nombre
- Recorrer el sistema de archivos

Visión jerárquica
Organización jerárquica en archivos y directorios



Concepto de directorio

- **Un directorio es un archivo**
  - un tipo abstracto de datos
  - el elemento necesario para organizar la información almacenada en memoria secundaria
- **Objetivos**
  - **Localiza**r rápidamente un **archivo** a partir del **nombre** asociado
  - Implementar un esquema de nombres conveniente para el usuario
  - El usuario puede establecer sus propias agrupaciones de archivos
  - **Protección**.- el propietario puede controlar las operaciones permitidas a cada usuario
    - Sobre directorios.- crear o borrar entradas, listar contenido, buscar

- **Operaciones sobre directorios**
  - El sistema operativo debe ofrecer un conjunto de llamada básicas para trabajar con directorios
    - **Crear entrada**
      - Requiere tener espacio libre para crearla
    - **Borrar entrada**
      - Libera el espacio en disco asociado al archivo y borra la entrada de directorio asociada ej. unlink(nombre)
    - **Buscar por nombre**
      - Normalmente la búsqueda se realiza de forma secuencial
    - **Listar contenido directorio**
      - Permite visualizar el contenido de los directorios
    - **Renombrar fichero**
      - Modifica la entrada a directorio
    - **Recorrer sistema de ficheros**
      - Permite situarse en cualquier punto de la jerarquía del directorio

- **Contenido**
  - Concepto de directorio
  - **Implementación de directorios**
  - Enlaces o referencias a archivos
  - Gestión del espacio de disco
  - Protección

- **Estructura de directorios**
  - **Directorio** -> mantiene la asociación de nombres a archivos

Estructura de directorios

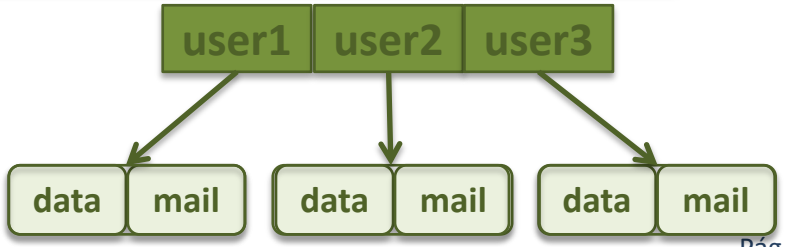
Plana

- Todos los archivos en un único directorio
  - Posibles colisiones entre nombres
  - No permite agrupar por usuarios/temas

Jerárquica

- Organización en niveles (árbol, grafo) con profundidad arbitraria
  - Permiten crear agrupaciones arbitrarias
  - Permiten montar/desmontar otros sistemas de archivos

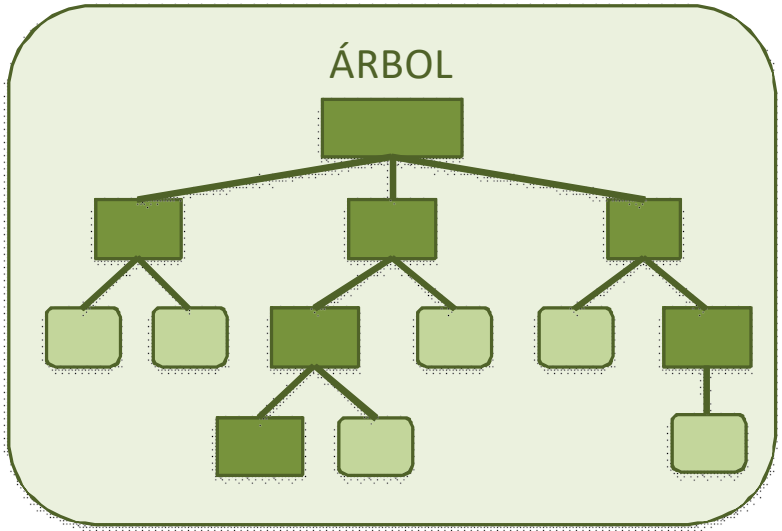
Directorio de un único nivel



- **Estructura jerárquica de directorios**

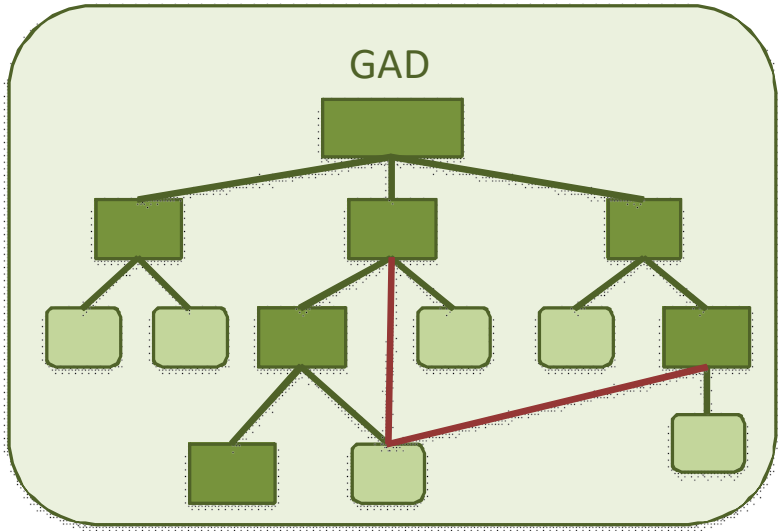
Árbol

- Búsqueda y agrupación eficientes
- Un único nombre absoluto para cada archivo = recorrido desde la raíz al nodo concreto
- Nombre relativo = recorrido desde directorio actual al nodo



Grafo Acíclico Dirigido (ej. Unix)

- Estructura básica en árbol, pero permite compartir ficheros y directorios
- Varios nombres absolutos (varias rutas desde la raíz) para un mismo nodo
- No permite ciclos

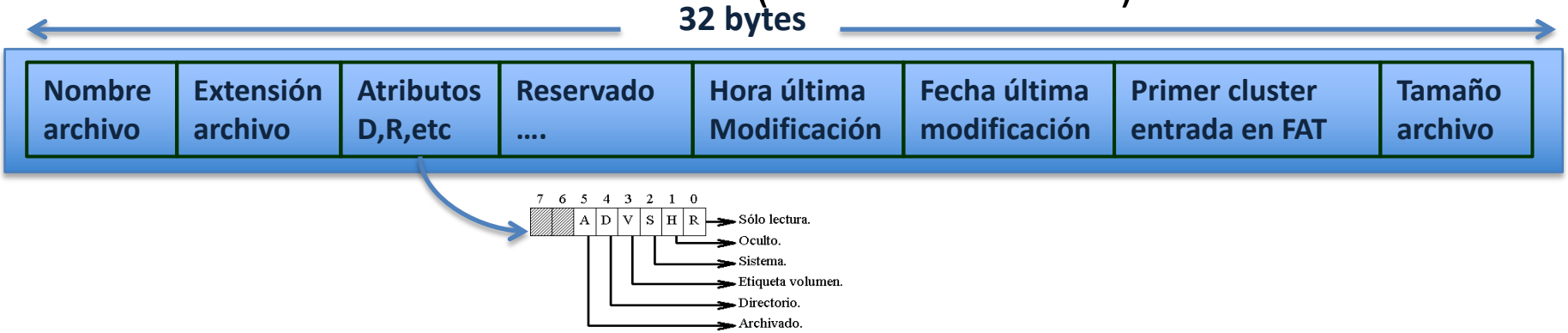


Implementación de directorios

- **Contenido de un directorio**
  - La información de un directorio esta organizada en registros
  - Un directorio contiene un conjunto de registros denominados **entradas de directorio**
  - Tiene una entrada por cada archivo existente en el directorio
- **Entrada de directorio**
  - **Contenido** de cada entrada es dependiente de la implementación del sistema de archivos
  - Nombre + referencia a los atributos (ej. Unix)
  - Nombre + atributos + referencia a datos (ej. Windows)
- **Ubicación** de los datos de directorios en disco
  - Centralizada (área dedicada en disco)
  - En archivos (adecuada para sistemas jerárquicos)

Implementación de directorios

- **Directorios en MS-DOS →FAT**
  - Características
    - Directorio raíz en lugar conocido y con talla constante
    - Resto de directorios se gestionan como un archivo
  - Un entrada de directorio ocupa 32 bytes
    - Nombre del fichero + extensión (8 bytes + 3bytes)
    - Permisos, tipo,etc.
    - Fecha ,hora, talla, etc.
    - Primer cluster de datos (entrada en la FAT)



Estructura de un disco MS-DOS

Implementación de directorios

- **Directorios en sistemas UNIX**
  - Directorios implementados como un tipo de **archivo**
  - Los datos de un directorio están estructurados en una tabla con dos columnas: número de inode y nombre

Entrada de directorio

Número nodo-i	Nombre de archivo

- Cada entrada de directorio corresponde a un archivo
- La entrada de nombre "." corresponde al directorio actual, apunta a si mismo
- La entrada de nombre ".." corresponde al directorio padre
- Número de inode = índice para localizar el inodes en el vector de nodos-i que almacena el sistema en el disco

Contenido de un directorio

Entrada del directorio raíz

Nodo-i	Nombre archivo
1	.
1	..
3	dev
4	bin

Implementación de directorios

- **Mecanismo de acceso por nombre**
  - En el disco existe una sección dedicada a almcenar un **vector de nodos-i** (nodos índice)
    - Nodo-i = atributos + localización de los datos del archivo
    - En el vector de nodos-i hay un nodo-i por archivo
  - Nombre absoluto:
    - La búsqueda del archivo se comienza desde el directorio raíz
    - El directorio raíz siempre tiene asignado un nodo-i fijo (MINIX nodo-i 1)
    - ejemplo: /a/b/c
  - Nombre relativo
    - La búsqueda del archivo comienza desde el directorio actual
    - Ejemplo: b/c

```
Mientras queden elementos
  Si se trata de un directorio
     Comprueba permisos,
     localiza elemento en ese directorio,
     obtiene nodo-i
Devuelve  nodo-i final
```

Nodo-i	Nombre archivo
1	.
1	..
3	dev
4	bin
6	a

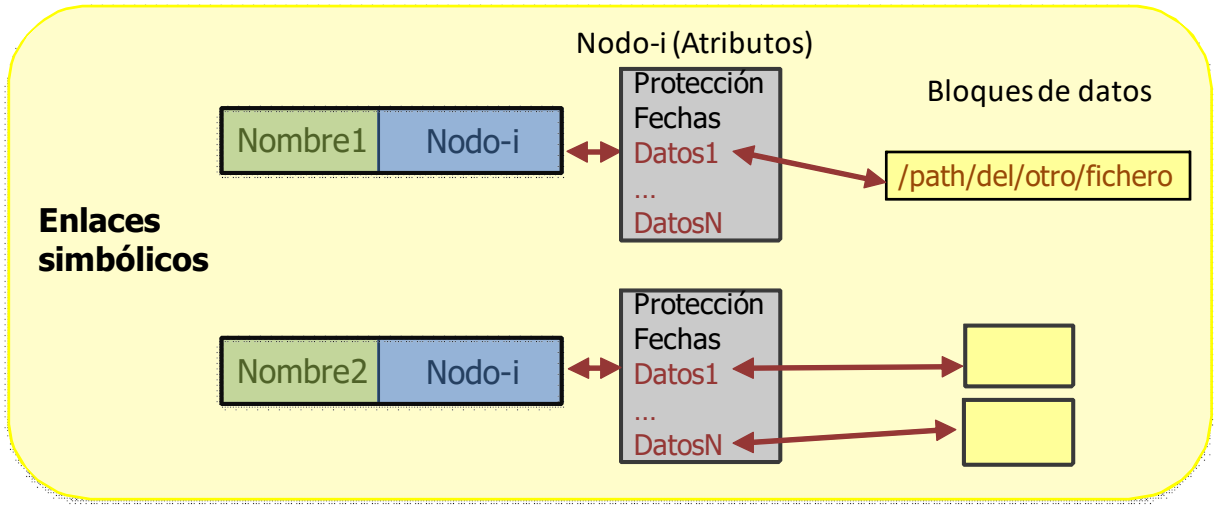
Nodo-i	Nombre archivo
6	.
1	..
10	b

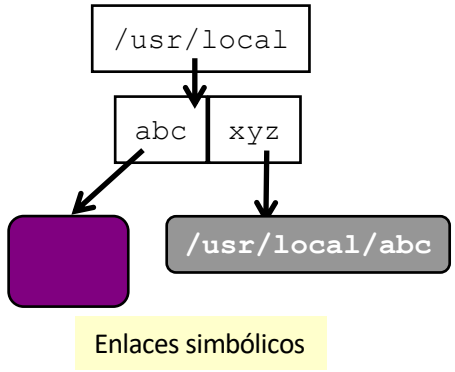
Nodo-i	Nombre archivo
10	.
6	..
20	c

Implementación de directorios

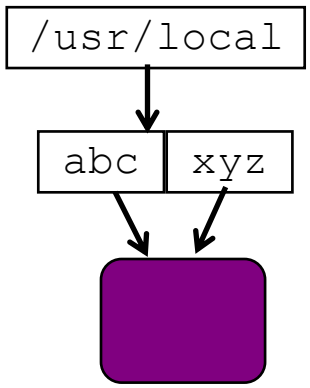
- **Contenido**
  - Concepto de directorio
  - Implementación de directorios
  - **Enlaces o referencias a archivos**
  - Gestión del espacio de disco
  - Protección

- **Enlaces Simbólicos**
  - En Unix se denominan **enlaces lógicos**, en Windows ficheros de **acceso directo**
  - A es un archivo de tipo 'enlace simbólico' que enlaza con el archivo B
    - Entre los atributos de A se indica el tipo del archivo (link)
    - El SO interpreta los datos de A como un path para acceder al archivo B
    - El SO redirige las lecturas y escrituras sobre A para que se acceda a los datos de B
    - A nivel de protección, se aplican los permisos del fichero B, no los de A
  - B puede estar en otro sistema de archivos (ej. montado remotamente)
  - ¿Qué pasa si el archivo B se borra o se desplaza a una nueva ubicación?
    - En algunos sistemas (ej MacOS) el propio SO corrige el path
    - En otros sistemas (ej Linux) el enlace deja de funcionar (queda huérfano)
  - Borrar el archivo enlace simbólico, borrar A, no afecta al referenciado (B)





- **Enlaces Físicos**
  - Dos o más entradas de directorio contienen el mismo número de inode
    - Un solo archivo al que se accede por varias rutas (varios nombres)
  - Cada **inode mantiene un contador** con el número de enlaces físico al archivo de dicho nodo-i
    - El archivo sólo se borra físicamente al eliminar su último nombre
  - Sólo resulta válido dentro de un mismo sistema de ficheros

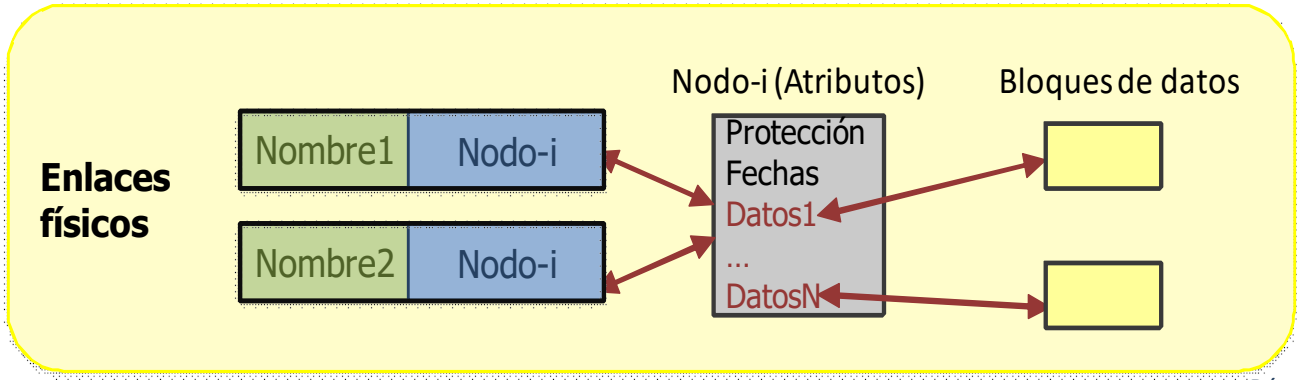


Enlaces físicos

inodo 10
....
Nº enlances=2
....
Pto triple ind.

Nodo-i	Nombre archivo
6	.
1	..
10	Nombre 1

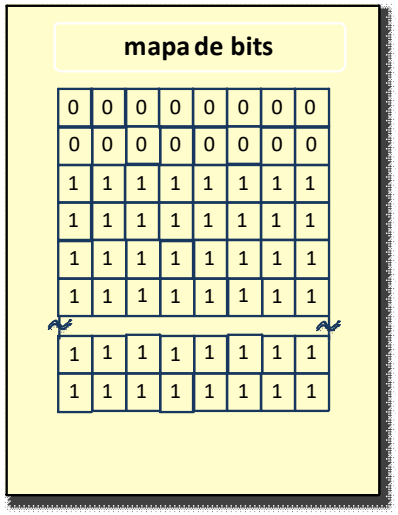
Nodo-i	Nombre archivo
20	.
5	..
10	Nombre 2

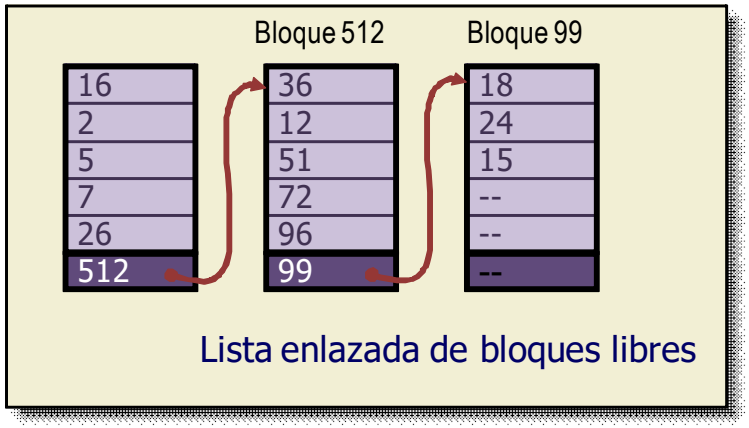


- **Contenido**
  - Concepto de directorio
  - Implementación de directorios
  - Enlaces o referencias a archivos
  - **Gestión del espacio de disco**
  - Protección

Gestión del espacio libre

- La gestión del espacio libre de **disco** ve al mismo como un **vector de bloques**
- En cada momento debemos saber **cuales están libres**
  - No nos sirve cualquier bloque
    - Buscamos **contigüidad** (por eficiencia posterior)
  - **Mapa de bits**
    - Cada bit representa un bloque en disco (ej. a 1 si bloque libre)
    - Se almacena en una zona dedicada de disco
    - Búsqueda eficiente de bloques libres consecutivos
  - **Lista enlazada**
    - En un lugar especifico del disco se mantiene el índice del primer bloque libre
    - Cada bloque libre apunta al siguiente
  - **Agrupamiento**
    - Se representan los bloques libres mediante una lista de bloques índice
    - Facilita insertar/extraer, complica la búsqueda bloques libres consecutivos

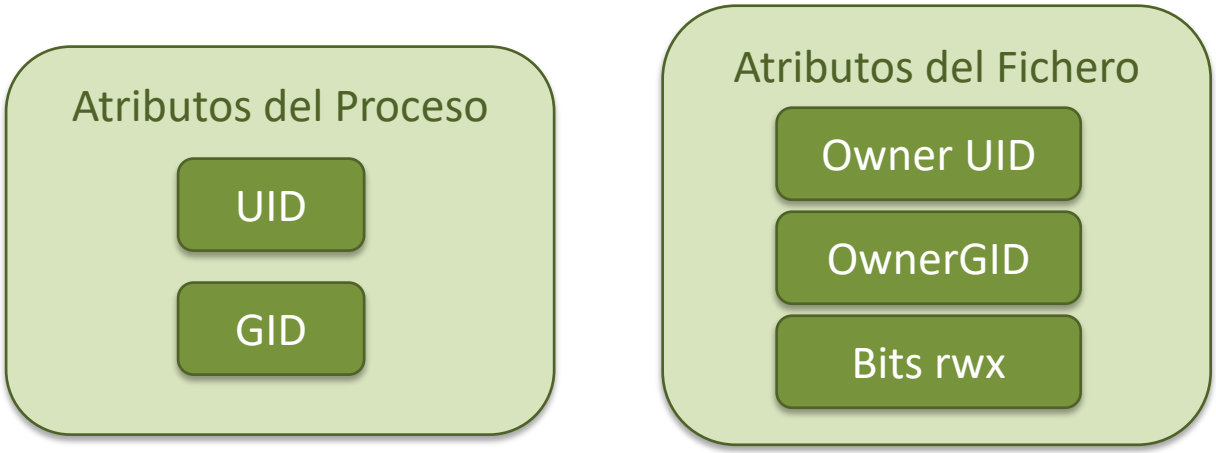




Implementación de directorios

- **Contenido**
  - Concepto de directorio
  - Implementación de directorios
  - Enlaces o referencias a archivos
  - Gestión del espacio de disco
  - **Protección**

- **Concepto de protección**
  - Mecanismo que se utiliza para controlar los accesos que los procesos realizan a los recursos del sistema
- **¿Como se realiza la protección en los sistemas UNIX?**
  - La protección en UNIX está basada en contrastar los atributos del proceso con los atributos del archivo y determinar si la operación se puede efectuar



- **Atributos de protección de proceso**
  - **Identificador de Usuario**
    - UID real (rUID) Identificador del usuario que ha creado el proceso
    - UID efectivo (eUID) Identificador del usuario para el que se ejecuta el proceso
  - **Identificador de grupo**
    - GID real (rGID) Identificador del grupo
    - GID efectivo (eGID) Identificador del grupo efectivo
- **Atributos de protección de archivo**
  - **Bits de permiso:** son 9 bits de permiso en grupos de tres (propietario, grupo, otros)
  - **Formatos :** rwxr_xr_x. 0755, 04755, rwsr_xr_x
  - Interpretación
    - **Archivos regulares**: lectura, escritura, ejecución
    - **Directorios:** listar contenido, crear o eliminar entradas, buscar
    - **Especiales:** lectura, escritura, ----------
  - BITS de SETUID, SETGID

- Asignación de atributos
  - El **fichero** recibe los atributos del proceso que lo crea

    `ownerUID = UID` `ownerGID = GID`
  - El **proceso** recibe los atributos gracias al mecanismo de herencia y a la información recogida en la tabla de usuarios (/etc/passwd)

    `nombre:contraseña:UID:GID:descripción:HOME:shell`
  - Un proceso puede cambiar de UID y GID cuando haga exec() sobre un fichero con el SETUID o SETGID activados
    - Si el ejecutable tiene el bit de SETUID activo, el eUID para a ser el del "ownerUID" del fichero
    - Si el ejecutable tiene el bit de SETGID activo, el eGID para a ser el del "ownerGID" del fichero
    - Ejemplo

```
-rwsr—r-x  1  felip users  17  Jan  29  09:34  arxi1
-rwxr-sr-x  1  felip users  223  Jan  29  09:34  arxi2
```

- **Reglas de protección en Unix**

  Cuando un proceso intenta realizar un acceso sobre un archivo, hay que comprobar las siguientes reglas en el orden que se indica

1) Si el proceso tiene **eUID=0** (superusuario), no se realiza ninguna comprobación y se admite el acceso

→ Excepto el permiso de ejecución que ha de estar establecido para algún dominio

2)Si el **proceso tiene eUID** igual al del propietario del archivo se comprueba la parte del propietario

3) Si no, si el **proceso tiene eGID** igual al del grupo del propietario del archivo, se comprueba la parte del grupo.

4) Si ninguna de las consultas anteriores ha funcionado se comprueba la parte del resto de usuarios (otros)

→Observe que si se comprueba un dominio no se realiza ninguna comprobación en el resto de dominios. (dominios: usuario, grupo, otros).

```
si UID = 0
entonces
   permiso de acceso concedido
sino
   si eUID = ownerUID
   entonces
     se concede permisos según
     la primera tripleta
                    → rws rwx r-x
   sino
      si eGID = ownerGID
      entonces
       se concede permisos según
       la segunda tripleta
                    → rws rwx r-x
      sino
       se concede permisos según
       a tercera tripleta
                    → rws rwx r-x
```

- **Ejemplo:**
- $ ls -l

```
total 7
-rwsr-xr-x   1 felip users      17 Jan 29 09:34 ejec1
-rwxr-sr-x   1 felip users     223 Jan 29 11:03 ejec2
-rw-------   1 felip users    5120 Jan 29 12:00 datos
```

  - El fichero datos sólo puede ser accedido (para lectura o escritura) por felip.
  - El fichero ejec1 tiene el bit SETUID fijado (s, en propietario) y permisos de ejecución para los otros dominios. Esto permite que cuando un proceso de otro dominio lo ejecute, pase al dominio del propietario (felip) y pueda leer/escribir el archivo datos.