# Gestión del Sistema de Archivos

*Dra. Carolina Mañoso*
*Dpto. Informática y Automática.UNED*

# Introducción

- Se necesitan tres condiciones para el almacenamiento de la información a largo plazo:
  - Se debe poder almacenar una cantidad grande de información
  - La información debe permanecer cuando el proceso termina
  - Debe ser posible que varios procesos tengan acceso concurrente a la información
- Solución: almacenamiento en un sistema auxiliar, en una estructura de archivos

# Índice

- El sistema de archivo desde el punto de vista del usuario
  - Forma de nombrar a los archivos
  - Operaciones permitidas sobre ellos
  - Directorios
  - Operaciones sobre ellos
- El sistema de archivo desde el punto de vista del diseñador
  - Forma de almacenamiento de los archivos y directorios
  - La administración del espacio en disco
- Seguridad y mecanismos de protección

# *Definición de archivo*

- El s.o. da una visión uniforme para todos los sistemas de almacenamiento, definiendo una unidad lógica de almacenamiento denominada **archivo**. Es función del s.o. el encargarse de asignar el contenido del archivo a espacios en el dispositivo físico

- Se considera como archivo a un conjunto de información relacionada definida por su creador, en general es una serie de bits, bytes o registros cuyo significado está definido por su autor y los usuarios

# *Nombre y atributos*

- Los archivos son nombrados y referenciados por su nombre. La forma de nombrar a los archivos cambia de un s.o. a otro

- Además del nombre, los archivos tienen otras propiedades como su tipo, la fecha y hora de su creación, el nombre o identificador del creador, su longitud y algunos más. A estas propiedades se les suelen denominar **atributos** y varían de un sistema a otro

# Tipos y estructuras (1/2)

- El s.o. puede tener conocimiento de los distintos tipos de archivos según las distintas estructuras lógicas que los formen, y así dar un mejor servicio

- Este conocimiento por parte del s.o. de los distintos tipos de archivos origina desventajas:
  - Mayor tamaño del s.o.
  - Sólo se pueden considerar tipos de archivos definidos por el sistema

- Unix sólo interpreta unos tipos especiales de archivos
  - Directorios (archivos del sistema para mantener la estructura del sist.)
  - Archivos especiales (para modelar periféricos)

# *Tipos y estructuras (2/2)*

- La estructura tanto lógica como física ha ido evolucionando:
  - Unix y Dos ven los archivos como bytes cuyo significado lo dan los programas de los usuarios
  - Otra estructura (antigua) es considerar los archivos como secuencias de registros lógicos
  - Otra estructura (de uso restringido) es arborescente

- **El acceso** a los elementos de un archivo puede ser:
  - Acceso **secuencial** (antiguo)
  - Acceso **arbitrario** (actual)

# Operaciones con archivos

- create (nombre_archivo, atributos)  (crear)
- open (nombre_archivo, modo_acceso) (abrir)
- seek (nombre_archivo, posicion_logica) (buscar )
- read (nombre_archivo, numero_bytes, buffer_entrada)
- write (nombre_archivo, numero_bytes, buffer_salida)
- close (nombre_archivo)
- delete (nombre_archivo)
- rename (nombre_archivo_antiguo, nombre_archivo_nuevo)
- atributos (nombre_archivo, atributos)
- copyfile (nombre_archivo_fuente, nombre_archivo_destino)

# Directorios de archivos (1/3)

- Los directorios son tablas simbólicas de archivos
- Una entrada típica de directorio puede contener la siguiente información:
  - Nombre, tipo, número de versión del archivo
  - Puntero de acceso al archivo, dirección de comienzo en el disco
  - Lista de atributos: tamaño, estructura, dueño, modos de protección...
- En muchos sistemas, la tabla del directorio está dividida en dos:
  - En una se mantienen los nombres de los archivos con un número de identificación, el cual da acceso a la otra
  - Que es donde se tiene el puntero de acceso al archivo y la lista de atributos

# Directorios de archivos (2/3)

- Los archivos de usuario y sistema disponibles están catalogados en directorios de archivos, que pueden ser:
  - Directorio de nivel único: contiene todos los archivos del sistema
  - Árbol de directorios (Jerárquico): los usuarios pueden agrupar los archivos relacionados en subdirectorios
    - Las entradas al directorio correspondiente tienen un atributo más que indica si esa entrada corresponde a un archivo o a un subdirectorio
- Los nombres de los caminos pueden ser:
  - Completos: comienza en el directorio raíz
  - Relativos: comienza en el directorio actual

# Directorios de archivos (3/3)

- Por cada directorio existen dos entradas especiales
  - "." es una entrada al propio directorio (un puntero a si mismo)
  - ".." es una entrada al directorio padre
- Se consideran los directorios como archivos ubicados en disco:
  - Los directorios son archivos que tienen una lista de todos los archivos
  - Para localizar el directorio raíz al arrancar el sistema se le debe colocar en una dirección conocida por el volumen desde el que se arranca el sistema

# Operaciones con directorios

- makedir (crear directorio)
- removedir (borrar directorio)
- opendir (abrir directorio)
- closedir (cerrar directorio)
- readdir (leer directorio)
- renamedir (cambiar de nombre al directorio)
- link (enlazar)
- unlink (desenlazar)

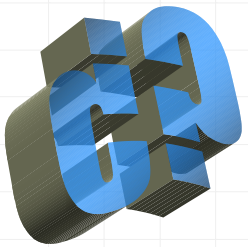
# Ejercicio 1

## Práctica de directorios

Indicar cinco nombres distintos de rutas de acceso para el archivo:

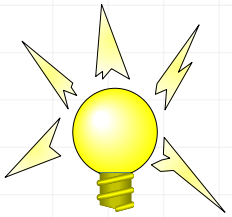
/usr/man/lista.

(considerar las entradas “.” y “..” del directorio)



# Solución 1

## Práctica de directorios



usr/./man/lista

usr/../usr/man/lista

usr/man/./lista

usr/man/../man/lista

usr/../usr/man/./lista

usr/../usr/man/../man/./lista

# *Gestión del espacio de disco*

- Gestión del espacio libre:
  - Mantener un **mapa de bits** de los bloques libres. Un disco con *n* bloques necesitará un mapa de bits con *n* bits. Los bloques libres se representarán con un 1 y los ocupados con un 0
  - Usar una **lista enlazada** de bloques libres, o una modificación de este método es utilizar una lista enlazada de bloques, en la que cada bloque contiene tantos números de bloques libres como pueda

- Método de asignación:
  - Asignación contigua
  - Asignación no contigua:
    - Listas enlazadas
    - Indexación

# Asignación contigua (1/2)

- Cada archivo ocupa un conjunto de direcciones contiguas en disco
- Las entradas en los directorios indicarán la dirección del primer bloque y la longitud del archivo
- La dificultad de este método es asignarle el espacio correcto cuando se crea el archivo. Si el archivo ocupa *n* bloques, se tiene que buscar *n* bloques contiguos libres:
  - método del primer ajuste (menos búsquedas)
  - método del mejor ajuste (reduce la fragmentación interna)

# *Asignación contigua: Inconvenientes (2/2)*

- No es realizable salvo que se conozca el tamaño máximo del archivo en el momento de su creación. (En algunas aplicaciones los archivos pueden crecer dinámicamente y no se conoce el tamaño) ⇒ Reubicación

- Fragmentación externa ⇒ Compactación

# *Listas enlazadas*

- Unos pocos bytes de cada bloque se dejan aparte para que apunten al siguiente bloque de la secuencia, el resto del bloque contiene los datos del archivo

- Las entradas en los directorios indicarán la dirección del primer bloque del disco asignado al archivo

- No se produce fragmentación externa
- Tampoco es necesario declarar el tamaño del archivo
- Los archivos puede crecer mientras haya bloques libres

- Adecuado para el acceso secuencial pero no para el aleatorio

# *Indexación*

- Se colocan los punteros (índices) a los bloques de los archivos en una tabla de índices

- Las entradas en los directorios indicarán la dirección del bloque donde están los índices a los bloques de datos del archivo

- Ventajas :
  - Ausencia de fragmentación externa
  - Eficacia tanto en el acceso aleatorio como en el secuencial

- Desventajas:
  - Pérdida de espacio dado el mantenimiento de las tablas de índices

# Compartición de archivos (1/2)

- **Link**: permite que un archivo o subdirectorio aparezca en varios directorios
  - Se específica el nombre del archivo y el camino de acceso, creándose un enlace entre este camino y el archivo ya existente
  - Unix mantiene una estructura de datos, nodo-i, asociada al archivo, de forma que los directorios apuntan al nodo-i correspondiente. El nodo-i mantiene un contador de los enlaces asociados al archivo

- Para la eliminación de un archivo compartido:
  - Si se elimina el archivo y el nodo-i, el segundo directorio tendrá una entrada a un nodo-i no válido
  - Si se elimina el archivo pero se mantiene el nodo-i, el dueño asignado al nodo-i sigue siendo el primero, hasta que se elimine del segundo directorio

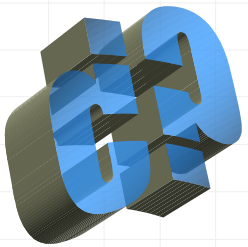
# Compartición de archivos (2/2)

- Solución: Enlaces simbólicos
  - Se crea un nuevo archivo que contiene la ruta de acceso al archivo al que se quiere enlazar
  - La entrada correspondiente del segundo directorio apuntará a este archivo de enlace.
  - De esta forma sólo el propietario verdadero del archivo tiene un apuntador al nodo-i.
  - Los usuarios enlazados al archivo tienen nombres de rutas de acceso y no apuntadores al nodo-i
  - Permite enlaces con archivos de otras máquinas
  - Inconvenientes: su alto coste en accesos a disco y las copias de seguridad pueden crear varias copias del mismo archivo

# Ejercicio 2

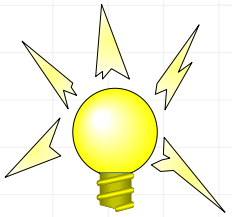
## Práctica de asignación de memoria en disco

El espacio libre del disco se puede gestionar mediante una lista de bloques o mediante un mapa de bits. Suponiendo que las direcciones del disco requieren *d* bits, que el disco tiene *b* bloques, de los cuales *l* están libres, deducir en que condiciones la lista de bloques libres utiliza menos espacio que el mapa de bits



# Solución 2

## Práctica de asignación de memoria en disco



- El tamaño que ocupa el mapa de bits será igual al número de bloques: $b$
- El tamaño que ocupa la lista de bloques será lo que ocupa cada dirección por el número de bloques libre: $d \times l$

Por lo tanto:

$$d \times l < b$$

# Ejercicio 3

## Práctica de asignación de memoria en disco

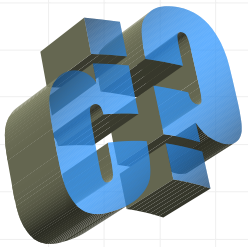
Calcular el número de accesos al disco que son necesarios para leer 30 bloques lógicos consecutivos de un archivo en un sistema con:

a) Asignación contigua

b) Asignación mediante listas enlazadas

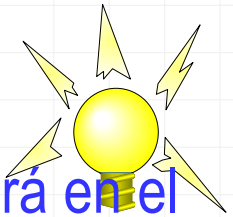
c) Asignación mediante indexación

.



# Solución 3

## Práctica de asignación de memoria en disco



a) Si el archivo ocupa 30 bloques, la asignación contigua comenzará en el bloque *b*, y ocupará hasta el *b*+30-1. Pero el acceso será de 1, el necesario para encontrar el primer bloque, *b*. De ahí que en este tipo de asignación basta con definir el primer bloque y la longitud del archivo.

b) Unos pocos bytes de cada bloque se reservan como puntero al siguiente bloque, el resto del bloque contiene los datos del archivo. Para leer un archivo sólo hay que seguir los punteros de bloque a bloque. Por lo tanto habrá que hacer tantos accesos como bloques, 30.

c) A parte de tener que ir a buscar los 30 bloques, habrá inicialmente que leer la tabla del disco y traerla a memoria, entonces, el número de accesos es de 30+1.

# Cachés de disco (1/2)

- Sirve para mantener un depósito de copias de bloques de disco más frecuentemente utilizados en la memoria principal de forma que permita eliminar accesos a disco repetidos satisfaciendo desde memoria las peticiones de lectura y/o escritura

- Se verifican todas las solicitudes de l/e para saber si el bloque solicitado se encuentra en la caché:
  - En caso afirmativo no hay que acceder al disco para satisfacer la petición
  - Si no, se tiene que leer el bloque para que se transfiera a la caché y se pueda satisfacer la demanda

# Cachés de disco (2/2)

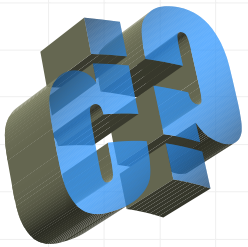
- Cuando se tiene que cargar un bloque en una caché totalmente ocupada ⇒ usar algoritmos de sustitución (FIFO, LRU...)

- En el caso de petición de escritura al disco, si la escritura se hace en la caché y posteriormente en el disco (escritura retardada) puede tener lugar inconsistencias:
  - Unix hace volcados periódicos (sinc)
  - MS-DOS, los bloques modificados se escriben inmediatamente en disco, escritura directa

# Ejercicio 4

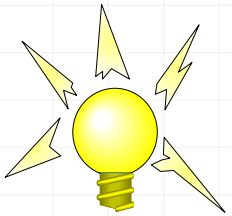
## Práctica de caché de disco

El rendimiento de un sistema de archivos depende de la razón de encuentros en la caché de disco (porcentaje de bloques disponibles en la caché frente al total solicitado). Si tarda 1 mseg en satisfacer una petición desde la caché y 40 mseg si se requiere la lectura de disco, ¿cuál es el tiempo promedio en satisfacer una solicitud si la razón de encuentros es *R*. (Considerar que *R* está comprendida entre 0 y 1). Representar gráficamente esta función



# Solución 4 (1/2)

## Práctica de caché de disco



- Si se tiene $R$ aciertos de caché, se tardará $R \times 1$ mseg en satisfacerlos.
- Entonces, $(1–R)$ no serán encuentros y se deberá hacer la lectura desde disco tardando:

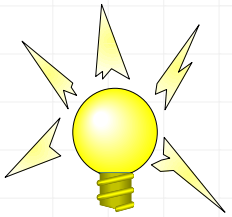
$$(1–R) \times 40 \text{ mseg}$$

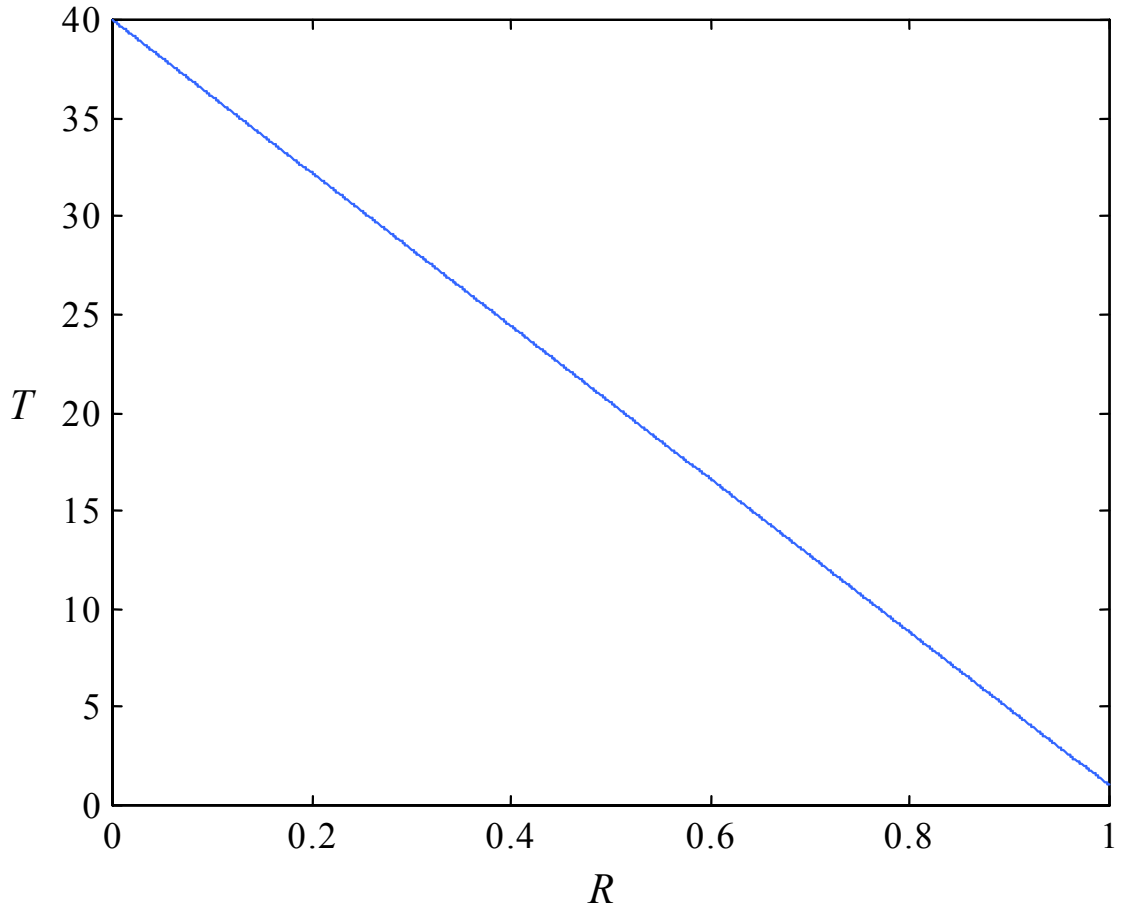
Así en total el tiempo será:

$$T = R + (1–R) \times 40$$

# Solución 4 (2/2)

## Práctica de caché de disco






# Seguridad y protección (1/2)

- Al almacenar información en un computador se debe proporcionar seguridad de la misma, proporcionando mecanismos (de protección) que la protejan tanto de daños físicos como de accesos inadecuados o mal intencionados

- Mantener la Integridad del sistema de archivos
  - Los problemas pueden venir por el uso de bloques que están en mal estado. Se solventa sabiendo cuales son los bloques defectuosos
  - Otras inconsistencias pueden ocurrir cuando el sistema falla en mitad de una operación de l/e/m de un bloque. Se solventa usando *fsck*

- Es necesario realizar copias de Seguridad:
  - Volcados periódicos del sistema de archivos
  - Volcados incrementales (complejidad en la restauración)

# Seguridad y protección (2/2)

- La penetración en un sistema informático se puede realizar:
  - La utilización por parte del intruso de la cuenta de un usuario legítimo
  - La ejecución de programas denominados *caballos de Troya*
  - La propagación de *gusanos y virus* informáticos
  - La inspección del sistema de archivos

- Identificación de usuarios:
  - Mediante contraseñas
  - Mediante artefactos
  - Mediante identificación física:
    - Fisiológica
    - De comportamiento

# *Mecanismos de protección y control de acceso (1/3)*

- Un proceso sólo debe poder acceder a aquellos recursos para los cuales está autorizado y que necesita en ese momento para completar su tarea. Este requisito se denomina Principio de la necesidad del saber

- Cada proceso trabaja dentro de un dominio, el cual especifica los recursos a los cuales puede tener acceso. Es el dominio de protección

- Cada dominio define un conjunto de objetos y las operaciones que se les pueden aplicar. La capacidad para ejecutar una operación sobre un objeto es un derecho de acceso

# *Mecanismos de protección y control de acceso (2/3)*

- Un dominio es un conjunto de derechos de acceso, cada uno de los cuales esta formado por un par de la forma:
  <nombre del objeto, conjunto de derechos>

- Las relaciones entre dominios y objetos se pueden representar de forma abstracta mediante una matriz denominada matriz de acceso. Las filas representan dominios y las columnas objetos
  - Esta matriz de acceso es una matriz dispersa con muchos elementos vacíos

- Por ello, una forma sencilla de representarla es mediante una tabla global consistente en tripletas ordenadas (dominio, objeto, derechos)
  - La tabla suele ser grande ⇒ no se puede conservar en memoria principal

# Mecanismos de protección y control de acceso (3/3)

- La matriz se suele almacenar:
  - Por columnas: lista de acceso. Cada objeto se le asocia una lista ordenada con todos los dominios que pueden tener acceso al objeto y la forma de dicho acceso
    - Retardo en la búsqueda para verificar el acceso. Se agrupan los usuarios en grupos (Unix)
  - Por filas: lista de capacidades. Cada dominio (o sujeto) se le asocia una lista de objetos a los cuales puede tener acceso, junto con una indicación de las operaciones permitidas sobre cada objeto
    - Revocación de acceso a un objeto