SISTEMAS OPERATIVOS:

Lección 12:
Directorios

Jesús Carretero Pérez
Alejandro Calderón Mateos
José Daniel García Sánchez
Francisco Javier García Blas
José Manuel Pérez Lobato
María Gregoria Casares Andrés

ADVERTENCIA

- Este material es un simple guión de la clase: no son los apuntes de la asignatura.

- El conocimiento exclusivo de este material no garantiza que el alumno pueda alcanzar los objetivos de la asignatura.

- Se recomienda que el alumno utilice los materiales complementarios propuestos.

Objetivos

- Conocer los conceptos de fichero y directorio así como sus características.
- Utilizar los servicios de gestión de Ficheros y directorios ofrecidos por el sistema operativo.
- Comprender la estructura de un sistema de ficheros.
- Comprender los mecanismos en los que se apoya un servidor de ficheros y aplicarlos a ejercicios sencillos.

Contenido

- Directorios.
- Alternativas de estructura.
- Interpretación de nombres.
- Manipulación de directorios.

Organización de ficheros

- Un sistema de ficheros puede almacenar gran cantidad de ficheros.

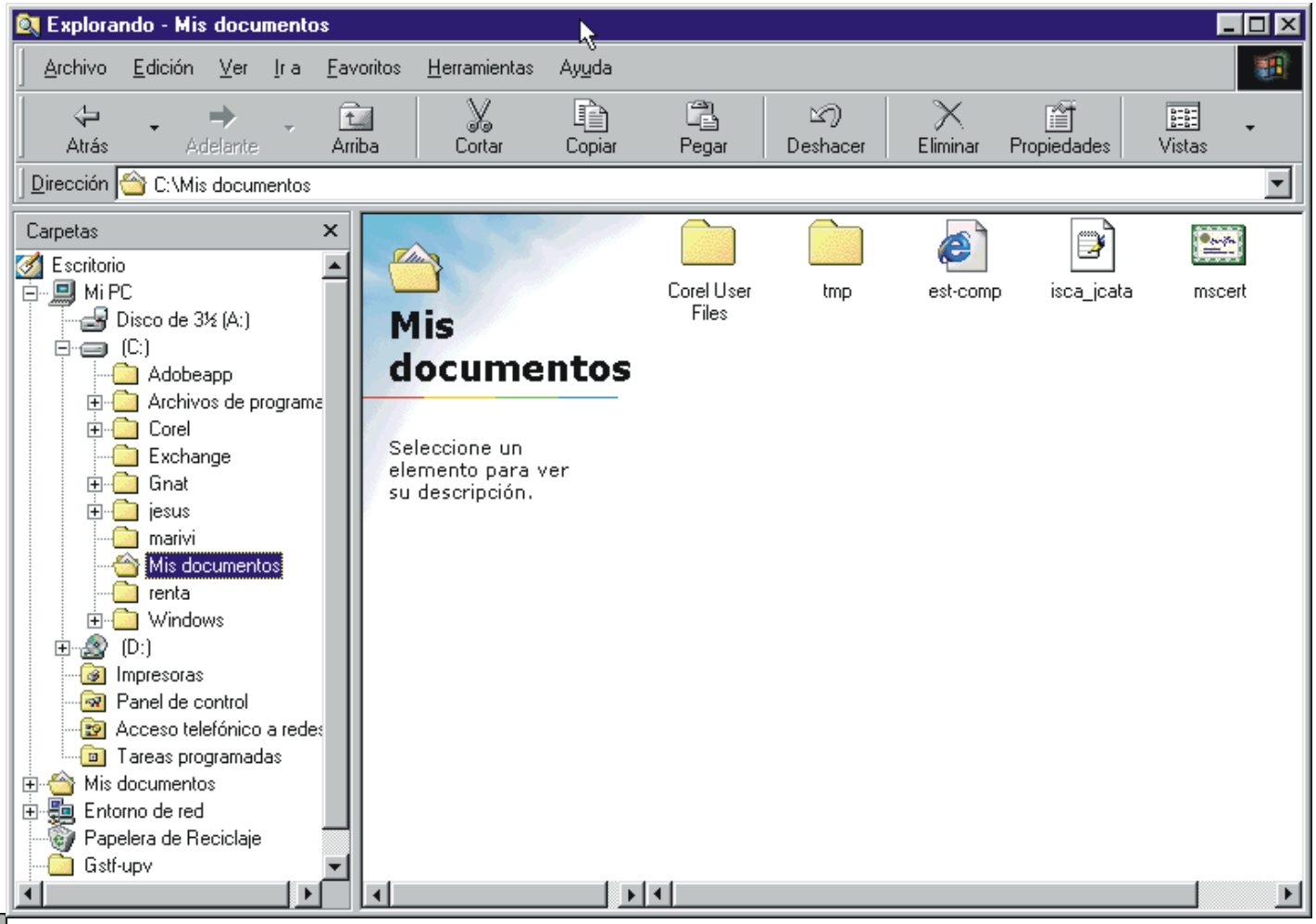
- Se necesita algún mecanismo para organizar y localizar los ficheros.

- Extensiones: Organización por tipo de fichero.
- Directorio: Metáfora de la carpeta con documentos.

Concepto de directorio

- **Directorio**: Objeto que relaciona de forma unívoca un nombre de usuario de fichero con su descriptor interno.
- Organizan y proporcionan información sobre la estructuración de los sistemas de ficheros.
- Un directorio tiene entrada por cada fichero que alberga.
- Información de la entrada:
  - Descriptor interno del fichero.
  - Posiblemente, algunos atributos del fichero.

Ejemplo: Explorador de Windows



Directorios: visión lógica

- Esquema jerárquico.
- Cuando se abre un fichero el SO busca el nombre en la estructura de directorios.
- Operaciones sobre un directorio:
  - Crear (insertar) y borrar (eliminar) directorios.
  - Abrir y cerrar directorios.
  - Renombrar directorios.
  - Leer entradas de un directorio.
- La organización jerárquica de un directorio
  - Simplifica el nombrado de ficheros (nombres únicos)
  - Proporciona una gestión de la distribución => agrupar ficheros de forma lógica (mismo usuario, misma aplicación)

Contenido

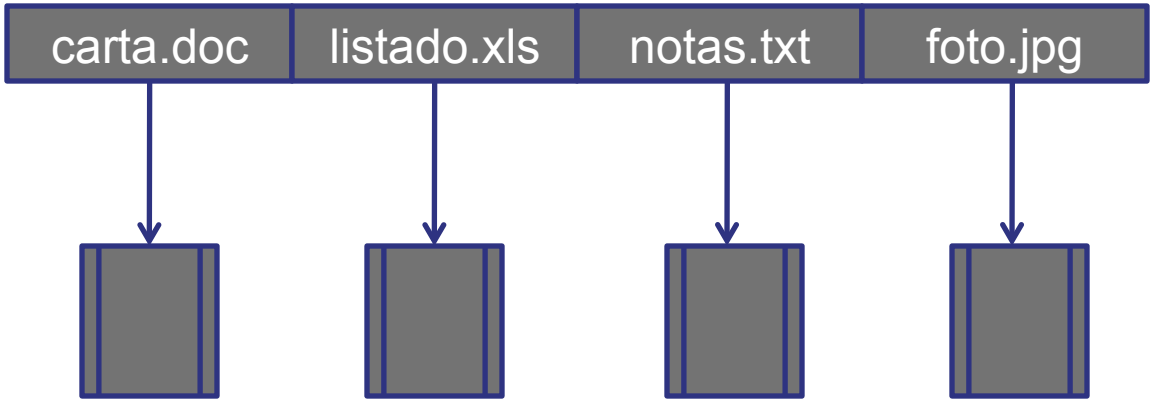
- Directorios.
- **Alternativas de estructura.**
- Interpretación de nombres.
- Manipulación de directorios.

Alternativas de estructura

- Directorio de un único nivel.
- Directorio de dos niveles.
- Directorio con estructura de árbol.
- Directorio con estructura de grafo acíclico.
- Directorio con forma de grafo generalizado.

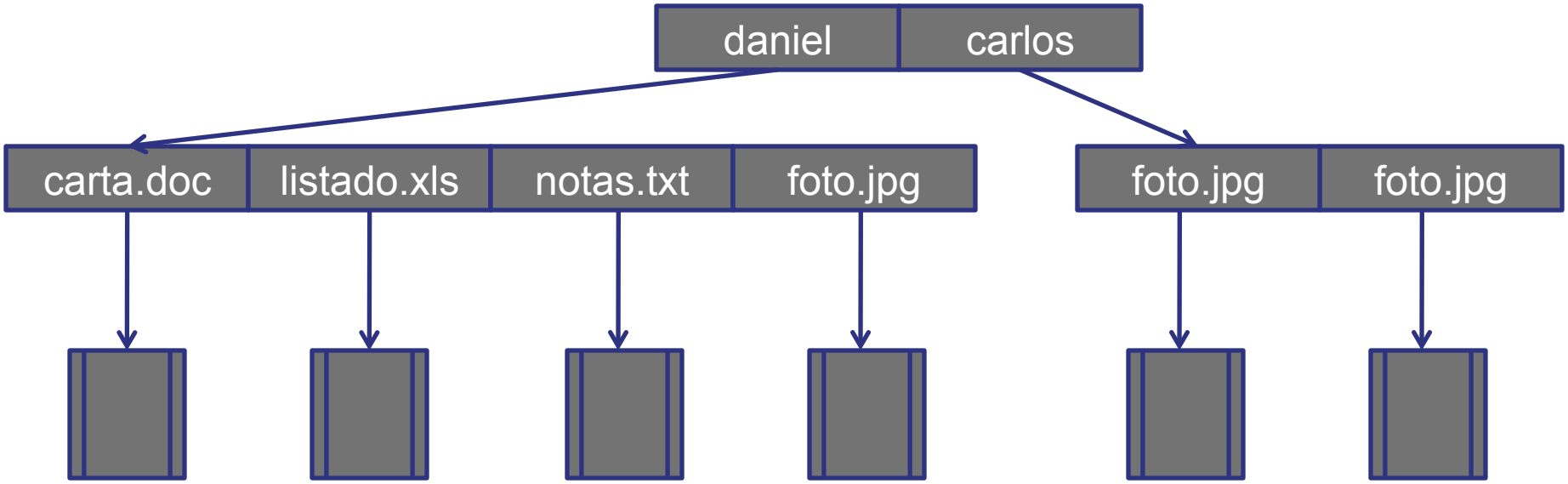
Directorio de un nivel

- Un único directorio para todos los usuarios.
- Problemas con el nombrado de los ficheros.
  - Alta probabilidad de coincidencia de nombres.



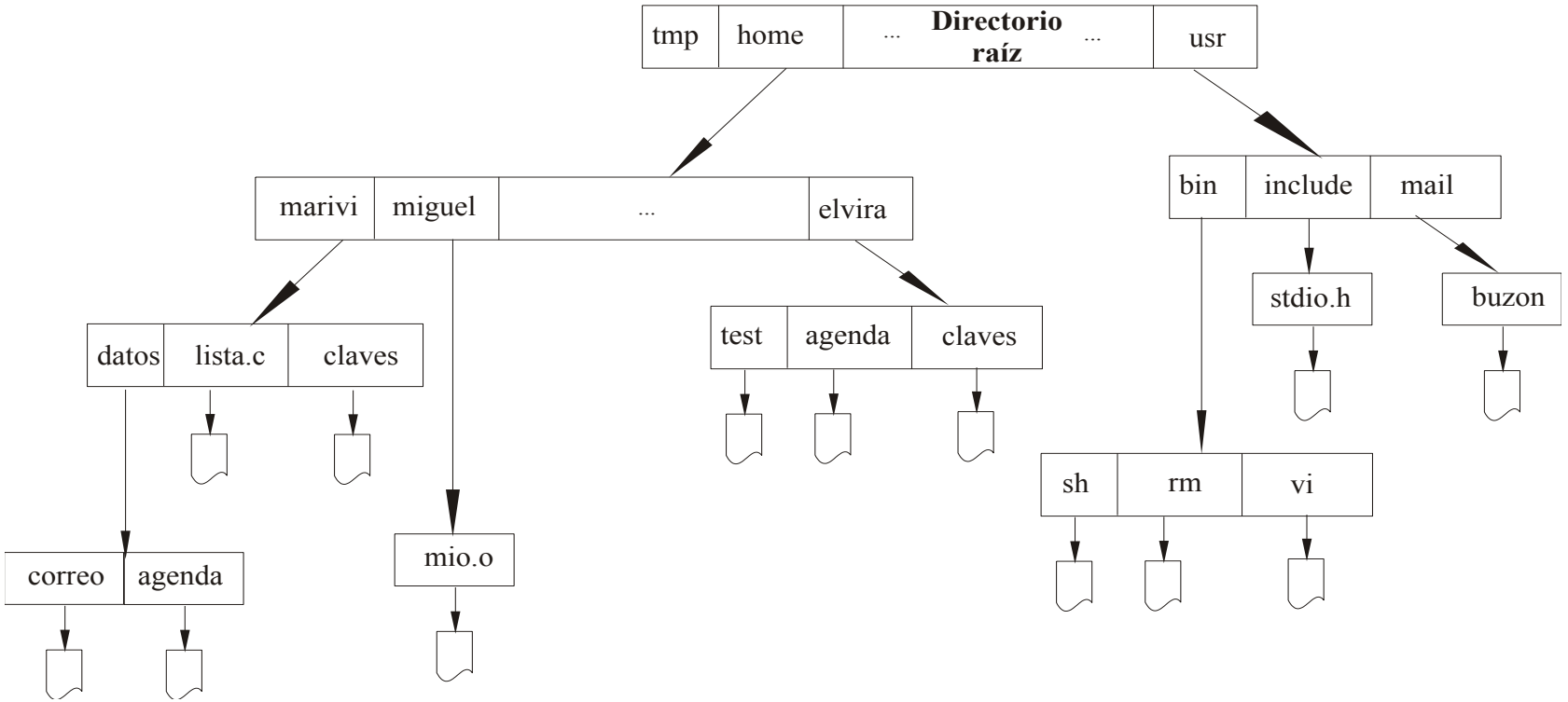
Directorio de dos niveles

- Un directorio por cada usuario.
- Camino de acceso automático o manual
- El mismo nombre de fichero para varios usuarios
- Búsqueda eficiente, pero problemas de agrupación



Directorio con estructura de árbol

- Búsqueda eficiente y agrupación
- Nombres relativos y absolutos -> directorio de trabajo

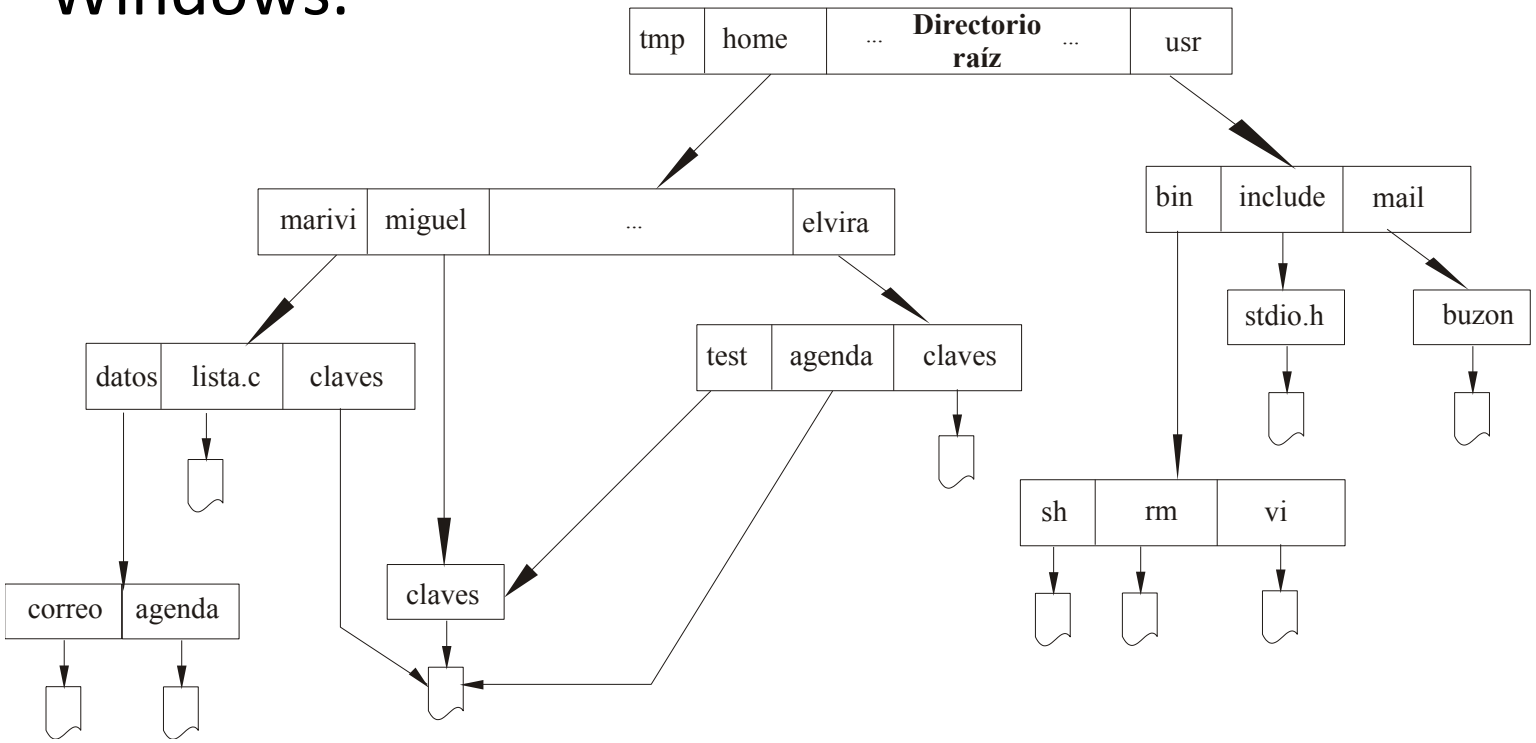


Directorio con estructura de árbol

- Los nombres absolutos contienen todo el camino
- Los nombres relativos parten del directorio de trabajo o actual
- Cambio de directorio:
  cd /spell/mail/prog
  cd prog
- Borrar un fichero: **rm** <nombre-fichero>
- Crear un subdirectorio: **mkdir** <nombre_dir>
- Ejemplo:
  cd /spell/mail
  mkdir count
  ls /spell/mail/count
- Borrar un subdirectorio: **rm** -r mail

Directorio de grafo acíclico

- Tienen ficheros y subdirectorios compartidos
- Este concepto no es visible para el usuario en Windows.



Directorio de grafo acíclico

- **link**: Un fichero con varios nombres -> control de enlaces
  - un único fichero con contador enlaces en descriptor (e. Físicos)
  - ficheros nuevos con el nombre destino dentro (e. simbólicos)
- Borrado de enlaces:
  - a) decrementar contador; si 0 borrar fichero
  - b) recorrer los enlaces y borrar todos
  - c) borrar únicamente el enlace y dejar los demás
- Problema grave: existencia de bucles en el árbol. Soluciones:
  - Permitir sólo enlaces a ficheros, no subdirectorios
  - Algoritmo de búsqueda de bucle cuando se hace un enlace
- Limitación de implementación en UNIX: sólo enlaces físicos dentro del mismo sistema de ficheros.

Estructura de los directorios

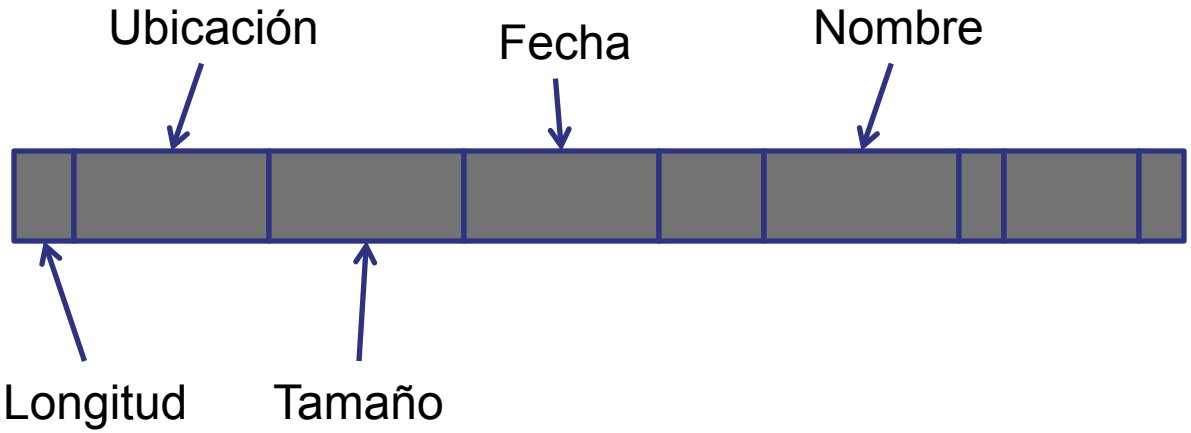
- Tanto la estructura del directorio como los ficheros residen en discos.
- Alternativas de implementación de directorios:
  - Utilizar bloques especiales con la información del directorio.
  - Utilizar un fichero cuyo contenido es el directorio.
- Información en un directorio: nombre, tipo, dirección, longitud máxima y actual, tiempos de acceso y modificación, dueño, etc.
  - En caso de usar un fichero la mayoría son metadatos de dicho fichero.

Estructura de directorio: Alternativas

- Directorios para ficheros contiguos.
  - Asumen que todos los ficheros se almacenan con asignación contigua.
- Directorios para ficheros enlazados.
  - Asumen que todos los ficheros se almacenan con asignación no contigua y los bloques se representan como una lista enlazada.
- Directorios para ficheros indexados.
  - Asumen que todos los ficheros se almacenan con asignación no contigua y los bloques o *extents* se representan mediante una estructura indexada

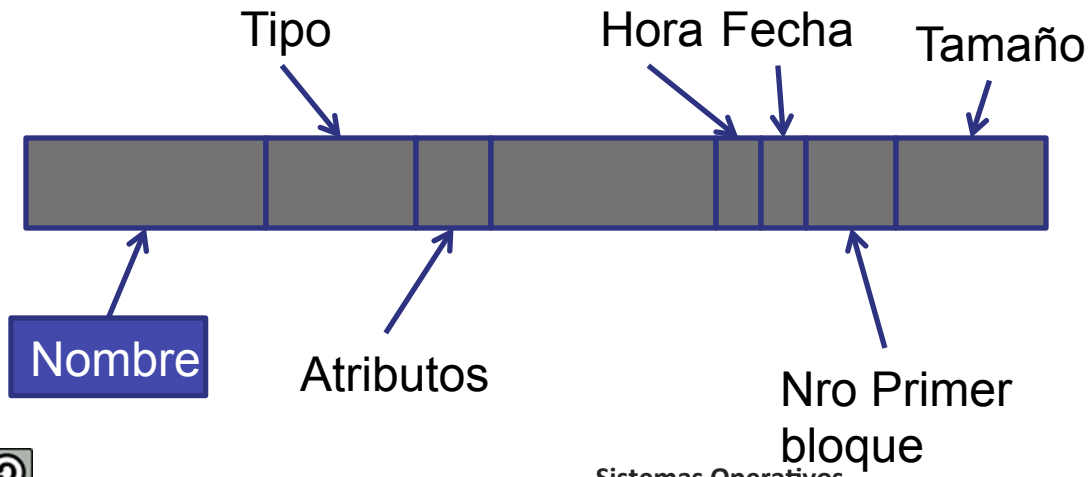
Directorio para ficheros contiguos

- Entrada de directorio:
  - Atributos del fichero en entrada de directorio.
  - Identificador del primer bloque del fichero.
  - Tamaño del fichero.
- Ejemplo: Formato ISO-9660 de CD-ROM



Directorio para ficheros enlazados

- Entrada de directorio:
  - Atributos de fichero.
  - Número del primer bloque.
  - Tamaño del fichero.
- Ejemplo: FAT



Directorio para ficheros indexados

- Alternativa más usada.
- Entrada de directorio:
  - Nombre.
  - Identificador de metadatos de fichero (nodo-i, entrada MFT, …).

Directorios para ficheros indexados

- Ventajas:
  - No hay que modificar el directorio para cambiar los atributos de un fichero.
  - No hay que modificar el directorio cuando un fichero cambia de longitud.
  - Un nodo-i puede representar un directorio o un fichero.
    - Sencillez en la construcción de sistemas jerárquicos.
  - La longitud de los nombres no está predeterminada.
  - Fácil creación de sinónimos para el nombre de un fichero.

Organización del directorio

- Eficiencia: localizar un fichero rápidamente
- Nombrado: conveniente y sencillo para los usuarios
  - Dos usuarios pueden tener el mismo nombre para ficheros distintos
  - Los mismos ficheros pueden tener nombres distintos
  - Nombres de longitud variable
- Agrupación: agrupación lógica de los ficheros según sus propiedades (por ejemplo: programas Pascal, juegos, etc.)
- Estructurado: operaciones claramente definidas y ocultación
- Sencillez: la entrada de directorio debe ser lo más sencilla posible.

Nombres jerárquicos

- Nombre absoluto: especificación del nombre respecto a la raíz (/ en LINUX, \ en Windows).
- Nombre relativo: especificación del nombre respecto a un directorio distinto del raíz
  - Ejemplo: (Estamos en /users/) miguel/claves
  - Relativos al dir. de trabajo o actual: aquel en el se está al indicar el nombre relativo. En Linux se obtiene con `pwd`
- Directorios especiales:
  - . Directorio de trabajo. Ejemplo: cp / users/miguel/claves .
  - .. Directorio *padre*. Ejemplo: ls ..
  - Directorio `HOME`: el directorio base del usuario

Contenido

- Directorios.
- Alternativas de estructura.
- **Interpretación de nombres.**
- Manipulación de directorios.

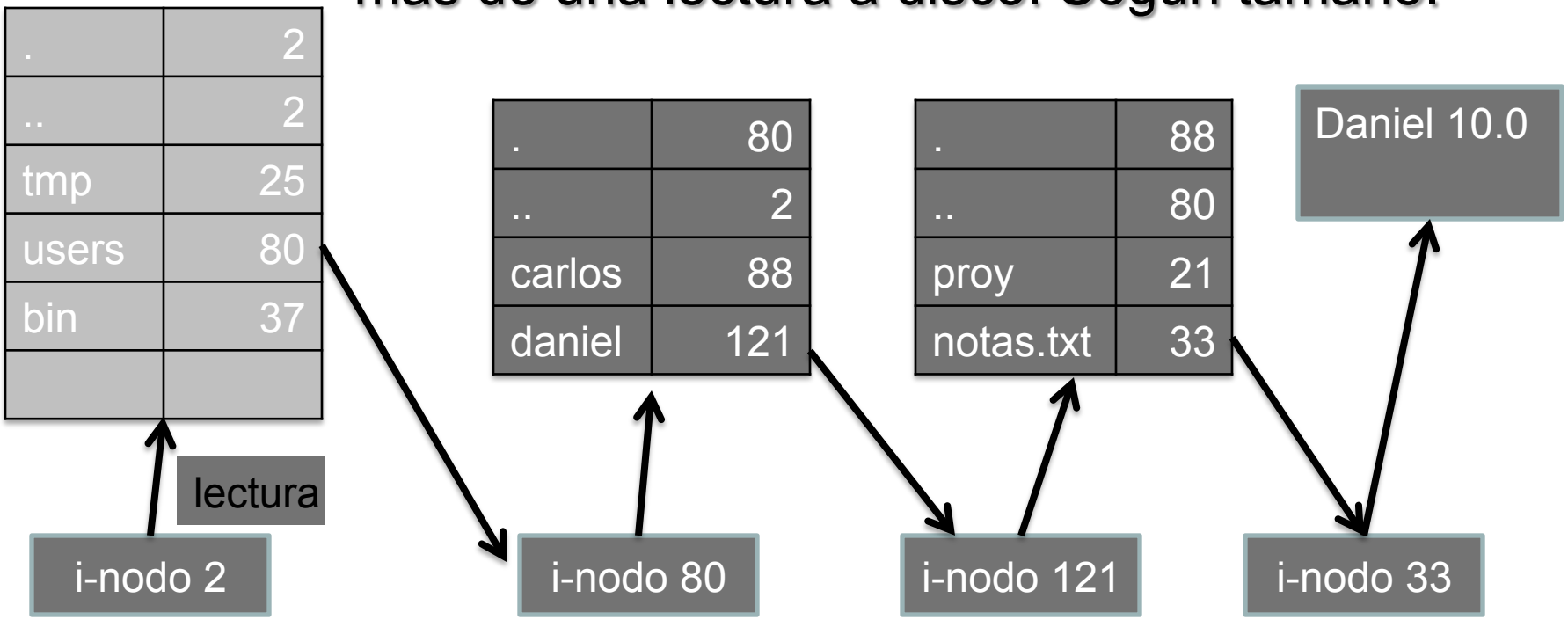
Interpretación de nombres en UNIX

- Cada directorio se almacena como un fichero con paras <número de i-nodo, nombre de fichero>.
- Inicialmente en memoria el directorio /.
- ¿Cuantos bloques de disco ocupa un directorio?
  - Depende del número de ficheros en el directorio y de la longitud de los nombres.
- La búsqueda en un directorio es secuencial.

Interpretación de nombres en UNIX

- Localizar el i-nodo del fichero /users/daniel/notas.txt.

El recorrido de cada directorio puede implicar más de una lectura a disco. Según tamaño.



Jerarquía de directorios

- ¿Árbol único de directorios?
  - Por dispositivo lógico en Windows (c:\users\miguel\claves, j:\pepe\tmp, ...)
  - Para todo el sistema en UNIX (/users/miguel/claves, /pepe/tmp, ...).
- Hacen falta servicios para construir la jerarquía: `mount` y `umount`.
  - `mount /dev/hda /users`
  - `umount /users`
- Ventajas: imagen única del sistema y ocultan el tipo de dispositivo
- Desventajas: complican la traducción de nombres, problemas para enlaces físicos entre ficheros

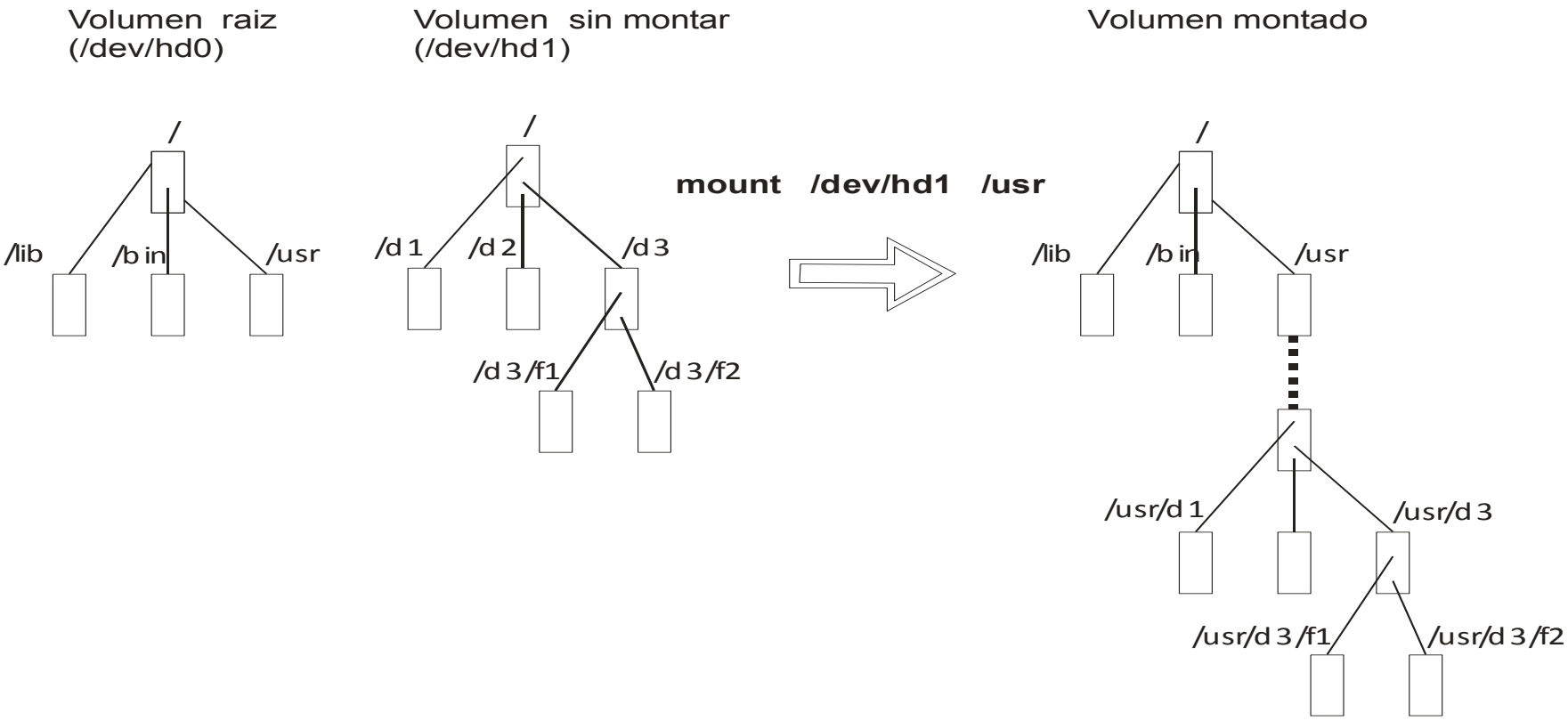
Sistemas de ficheros y particiones

- Volumen: conjunto coherente de metainformación y datos.
- Ejemplos de Sistemas de ficheros:

MS-DOS

UNIX

Montado de sistemas de ficheros o particiones



Contenido

- Directorios.
- Alternativas de estructura.
- Interpretación de nombres.
- **Manipulación de directorios.**

Ejemplo: manipulación de directorios

- Servicios que realizan el tratamiento de los archivos que representan directorios.
- ¿Cómo se sabe si un nombre corresponde con fichero o directorio?
- Servicio

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(char *name, struct stat *buf);
int fstat(int fd, struct stat *buf);
...
S_ISDIR(s.st_mode) /* cierto si se trata de directorio */
```

Mkdir – Crear un directorio

- Servicio:

```
#include <sys/types.h>
#include <dirent.h>
int mkdir(const char *name, mode_t mode);
```

- Argumentos:
  - `name` nombre del directorio
  - `mode` bits de protección
- Devuelve:
  - Cero ó -1 si error
- Descripción:
  - Crea un directorio de nombre `name`.
  - UID_dueño = UID_efectivo
  - GID_dueño = GID_efectivo

Rmdir – Borrar un directorio

- Servicio:

```
#include <sys/types.h>
int rmdir(const char *name);
```

- Argumentos:
  - `name` nombre del directorio
- Devuelve:
  - Cero ó -1 si error
- Descripción:
  - Borra el directorio si está vacío.
  - Si el directorio no está vacío no se borra.

Opendir – Abrir un directorio

- Servicio:

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir(char *dirname);
```

- Argumentos:
  - `dirname` puntero al nombre del directorio
- Devuelve:
  - Un puntero para utilizarse en `readdir()` o `closedir()`. `NULL` si hubo error.
- Descripción:
  - Abre un directorio como una secuencia de entradas. Se coloca en el primer elemento.

Closedir – Cerrar un directorio

- Servicio:

```
#include <sys/types.h>
#include <dirent.h>
int closedir(DIR *dirp);
```

- Argumentos:
  - `dirp` puntero devuelto por `opendir()`.
- Devuelve:
  - Cero ó -1 si error.
- Descripción:
  - Cierra la asociación entre `dirp` y la secuencia de entradas de directorio.

Readdir – Lectura de entradas de directorio

- Servicio:

```
#include <sys/types.h>
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
```

- Argumentos:
  - `dirp` puntero retornado por `opendir()`.
- Devuelve:
  - Un puntero a un objeto del tipo `struct dirent` que representa una entrada de directorio o `NULL` si hubo error.
- Descripción:
  - Devuelve la siguiente entrada del directorio asociado a `dirp`.
  - Avanza el puntero a la siguiente entrada.
  - La estructura es dependiente de la implementación. Debería asumirse que tan solo se obtiene un miembro: `char *d_name`.

Rewindir – Posicionar el puntero de directorio

- Servicio:

```
#include <sys/types.h>
#include <dirent.h>
void rewindir(DIR *dirp);
```

- Argumentos:
  - `dirp` puntero devuelto por `opendir()`
  - Descripción:
  - Sitúa el puntero de posición dentro del directorio en la primera entrada.

Link – Crear una entrada de directorio

- Servicio:

```
#include <unistd.h>
    int link(const char *existing, const char *new);
    int symlink(const char *existing, const char *new);
```

- Argumentos:
  - `existing` nombre del archivo existente.
  - `new` nombre de la nueva entrada que será un enlace al archivo existente.
- Devuelve:
  - Cero ó -1 si error.
- Descripción:
  - Crea un nuevo enlace, físico o simbólico, para un archivo existente.
  - El sistema no registra cuál es el enlace original.
  - `existing` no debe ser el nombre de un directorio salvo que se tenga privilegio suficiente y la implementación soporte el enlace de directorios

Unlink – Eliminación de entrad de directorio

- Servicio:

```
#include <sys/types>
int unlink(char *name);
```

- Argumentos:
  - `name` nombre de archivo
- Devuelve:
  - Cero ó -1 si error
- Descripción:
  - Elimina la entrada de directorio y decrementa el número de enlaces del archivo correspondiente.
  - Cuando el número de enlaces es igual a cero y ningún proceso lo mantiene abierto, se libera el espacio ocupado por el archivo y el archivo deja de ser accesible.

Chdir – Cambiar el directorio actual

- Servicio:

```
int chdir(char *name);
```

- Argumentos:
  - `name` nombre de un directorio
- Devuelve:
  - Cero ó -1 si error
- Descripción:
  - Modifica el directorio actual, aquel a partir del cual se forman los nombre relativos.

Rename – Cambiar el nombre de un archivo

- Servicio:

```
#include <unistd.h>
int rename(char *old, char *new);
```

- Argumentos:
  - `old` nombre de un archivo existente
  - `new` nuevo nombre del archivo
- Devuelve:
  - Cero ó -1 si error
- Descripción:
  - Cambia el nombre del archivo `old`. El nuevo nombre es `new`.

Getcwd – Obtener el nombre del directorio actual

- Servicio:

```
char *getcwd(char *buf, size_t size);
```

- Argumentos:
  - `buf` puntero al espacio donde almacenar el nombre del directorio actual
  - `size` longitud en bytes de dicho espacio
- Devuelve:
  - Puntero a `buf` o `NULL` si error.
- Descripción:
  - Obtiene el nombre del directorio actual

Ejemplo: Listado de un directorio

```
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>

#define MAX_BUF   256

void main(int argc, char **argv) {
  DIR *dirp;
  struct dirent *dp;
  char buf[MAX_BUF];

  /* imrpime el directorio actual */
  getcwd(buf, MAX_BUF);
  printf("Directorio actual: %s\n", buf);
```

Ejemplo: Listado de un directorio

```
  /* abre el directorio pasado como argumento */
  dirp = opendir(argv[1]);

  if (dirp == NULL)  {
    fprintf(stderr,"No puedo abrir %s\n", argv[1]);
  }
  else {
    /* lee entrada a entrada */
    while ( (dp = readdir(dirp)) != NULL)
      printf("%s\n", dp->d_name);
    closedir(dirp);
  }
  exit(0);
}
```

Lecturas recomendadas

- Básica

- Carretero 2007:
  - 9.3. Directorios
  - 9.4. Nombre jerárquicos.
  - 9.9. Estructura y almacenamiento del directorio.
  - 9.10. El servidor de directorios.

- Complementaria

- Stallings 2005:
  - 12.3 Directorios.
- Silberschatz 2006:
  - 10.3 Estructura de directorios.
  - 10.4 Montaje de sistemas de archivos.
  - 11.3. Implementación de directorios.

SISTEMAS OPERATIVOS:

Lección 12:
Directorios