Sistemas Operativos

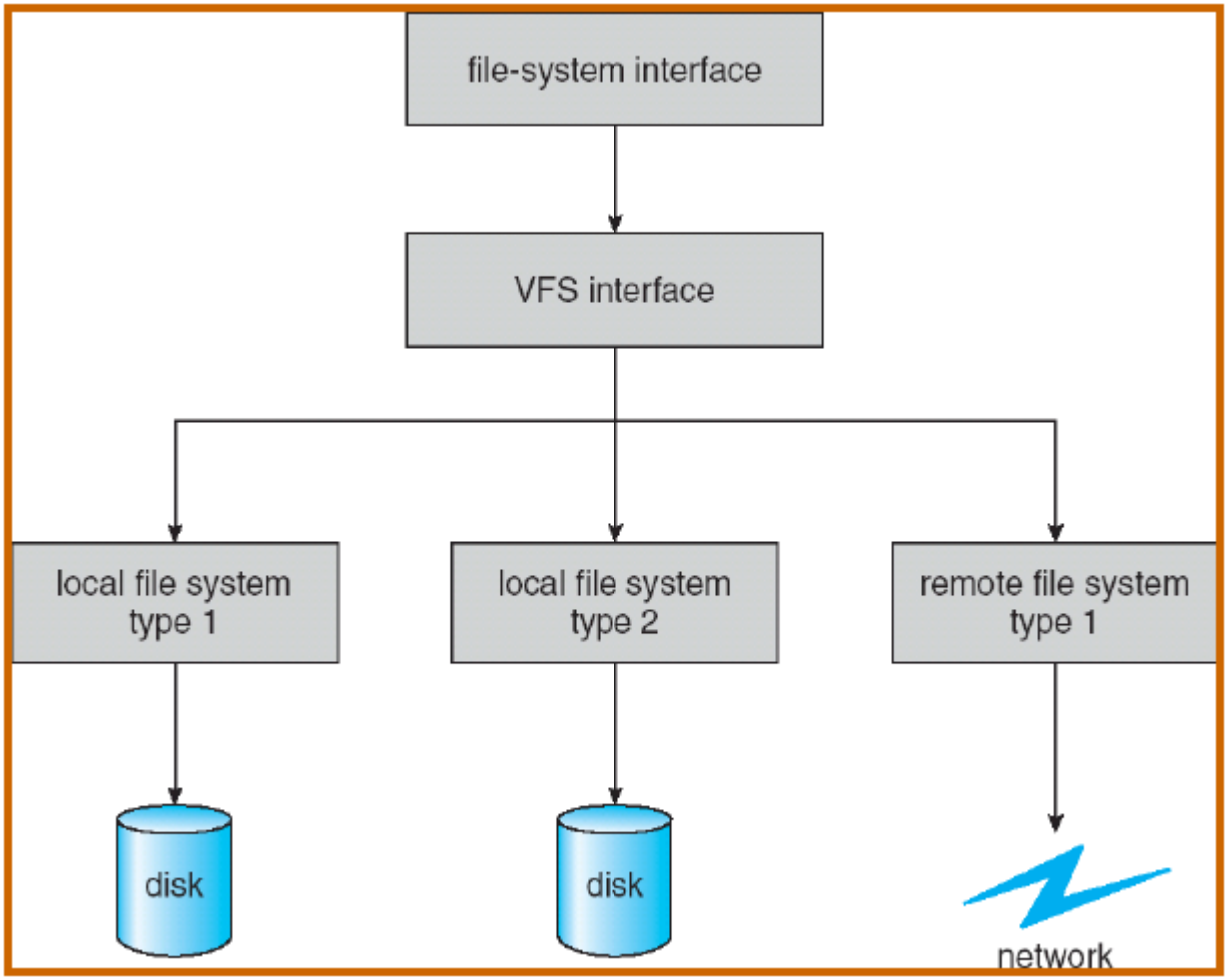
Curso 2016

Sistema de Archivos II

Sistema de archivos virtual

- Es común que un sistema operativo se acceda a más de una implementación de sistema de archivos (*ufs*, *ext2*, *ext3*, *jfs*, *jfs2*, *ntfs*, etc.).
- Se utilizan técnicas de orientación a objetos para lograr mantener una estructura independiente del sistema de archivos que se utilice.
- Se genera una estructura en tres capas:
  - Interfaz del sistema de archivo (*open*, *read*, etc.).
  - Sistema de archivos virtual (*Virtual File Server*).
  - Implementación especifica del sistema de archivo.

Sistema de archivos virtual



Sistema de archivos virtual

- El sistema de archivos virtual provee de dos funcionalidades importantes:
  - Propone una interfaz genérica de sistema de archivo que es independiente del tipo de sistema de archivo. De esta forma, se logra un acceso transparente al sistema de archivos.
  - Propone un bloque de control de archivo virtual que puede representar tanto archivos locales como remotos.

Estructura de los directorios

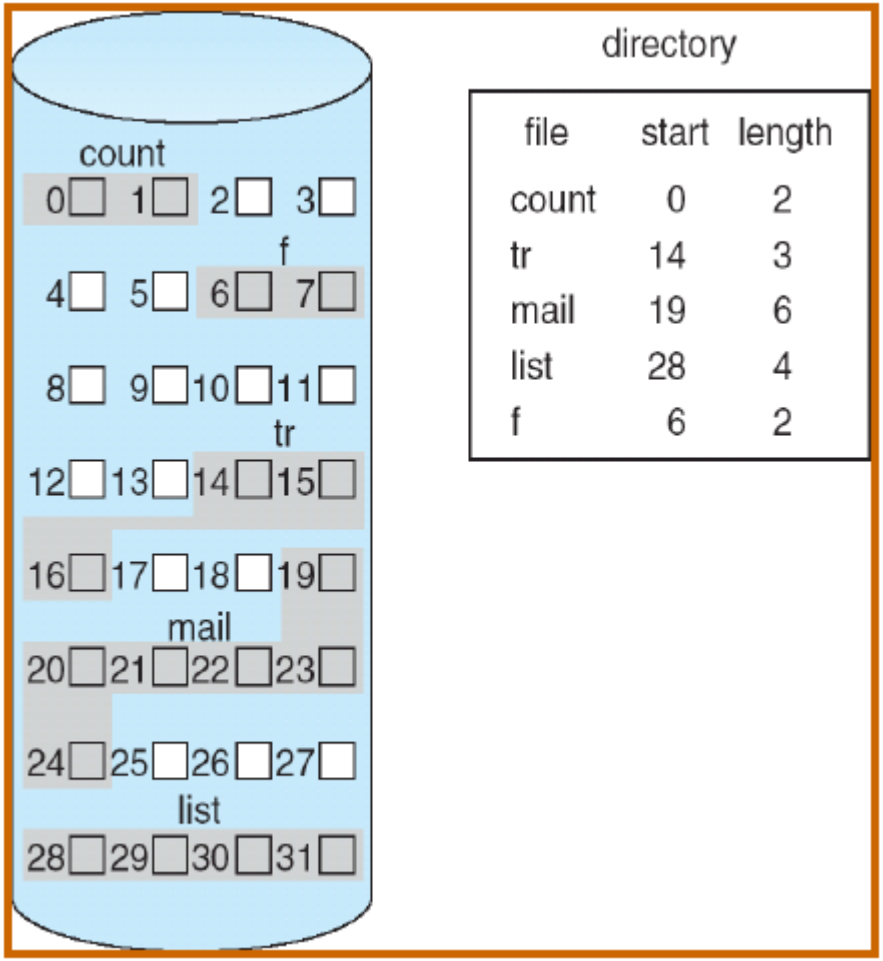
- Los directorios contienen la información de los archivos que pertenecen a él. Para organizar la información existen varias alternativas:
  - **Lista encadenada**: los nombres de los archivos y un puntero al bloque de control son dispuestos en una lista encadenada. En la búsqueda, inserción, borrado es necesario un acceso lineal. Es usual el uso de caches en memoria principal para acelerar el acceso.
  - **Tabla de *hash* abierto**: con el nombre del archivo se genera la clave utilizada.

Métodos de asignación

- Para la disposición de los datos de los archivos en disco se tienen, en general, tres métodos:
  - **Asignación contigua** (*Contiguous Allocation*): Los datos son dispuestos en forma contigua. Para mantener la información es necesario saber en que bloque comienza y la cantidad de bloques que tiene el archivo.
  - **Asignación en forma de lista** (*Linked Allocation*): Los bloques de datos forman una lista encadenada. Es necesario una referencia al primer y último bloque de datos en el bloque de control de archivo.
  - **Asignación indexada** (*Indexed Allocation*): Se mantiene una tabla en donde cada entrada referencia a un bloque de datos.

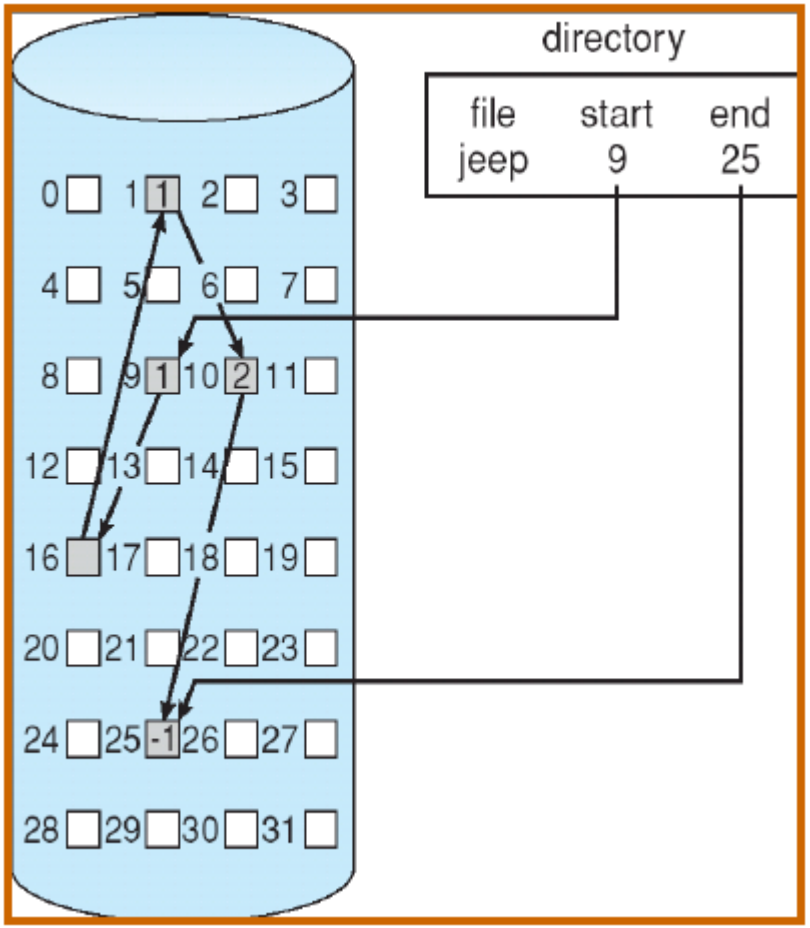
Asignación contigua (*Contiguous Allocation*)

- Sufre de fragmentación externa.
- Es necesario reubicar continuamente los archivos si crecen en tamaño.
- Se utilizan técnicas de asignación de tamaños más grandes para prever el crecimiento futuro de los archivos.



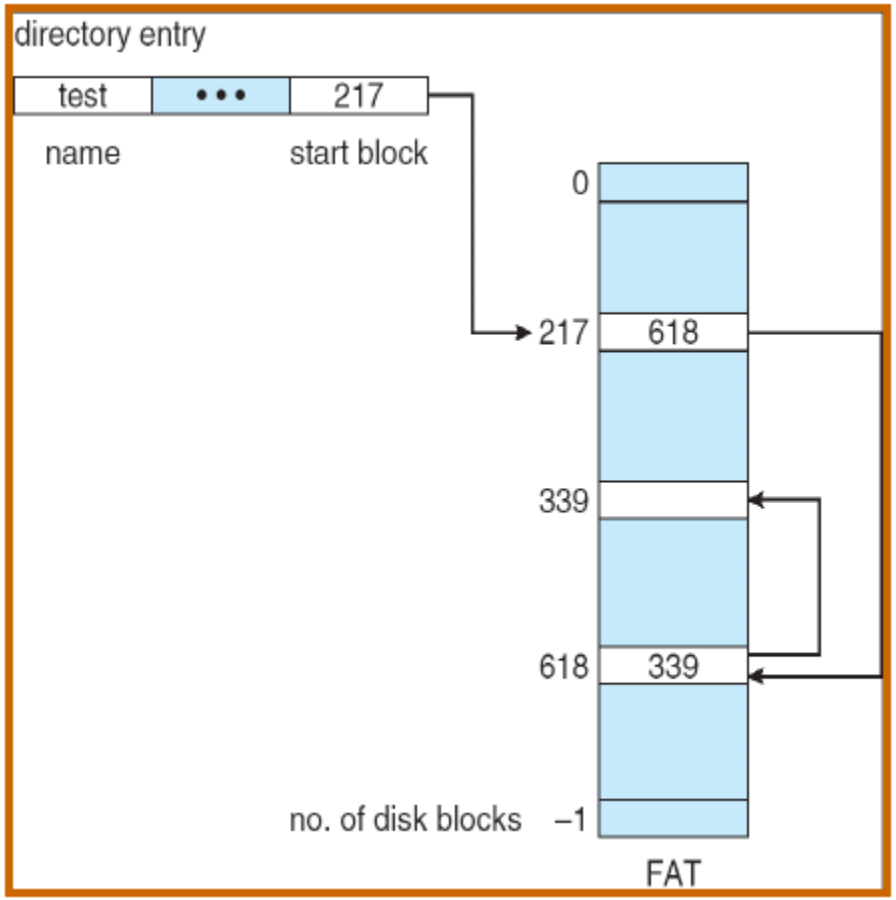
Asignación en forma de lista (*Linked Allocation*)

- Soluciona el problema de la fragmentación.
- El acceso a los bloques es lineal.
- Los punteros ocupan espacio en los bloques.
- La pérdida de una referencia genera la pérdida de gran parte de información del archivo.



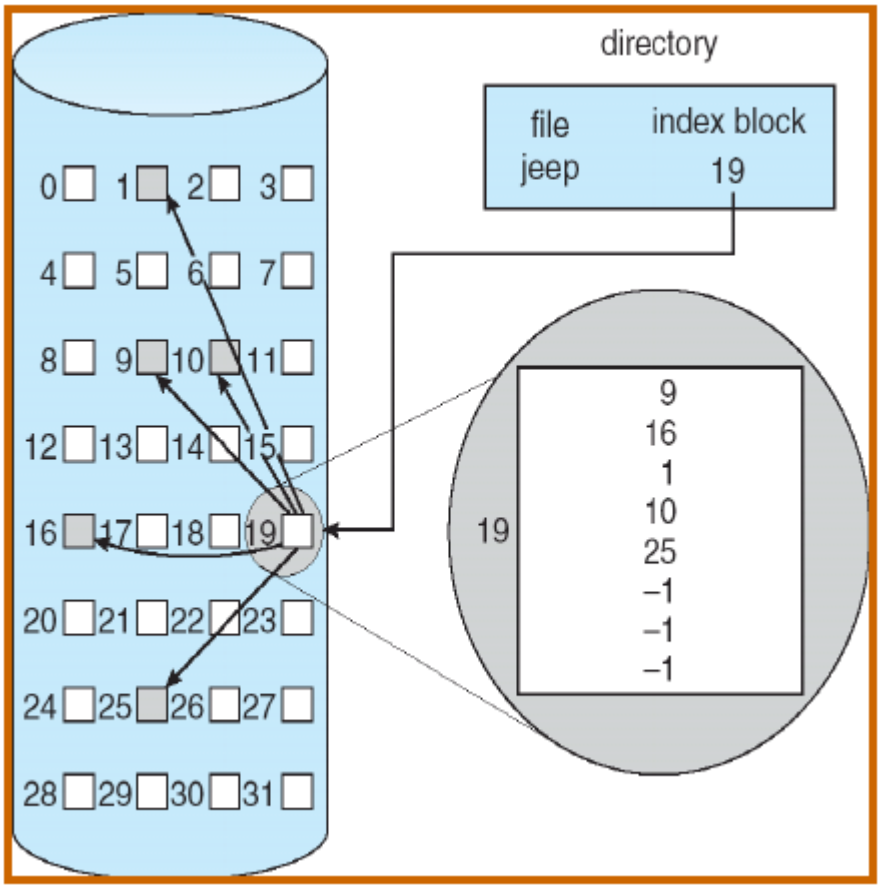
Asignación en forma de lista (*Linked Allocation*)

- Ej.: FAT
- Al comienzo de cada partición existe una tabla de asignación de archivos (*File Allocation Table*), que contiene la lista de bloques.
- La tabla tiene una entrada por cada bloque de disco, y es indexada por el número de bloque.



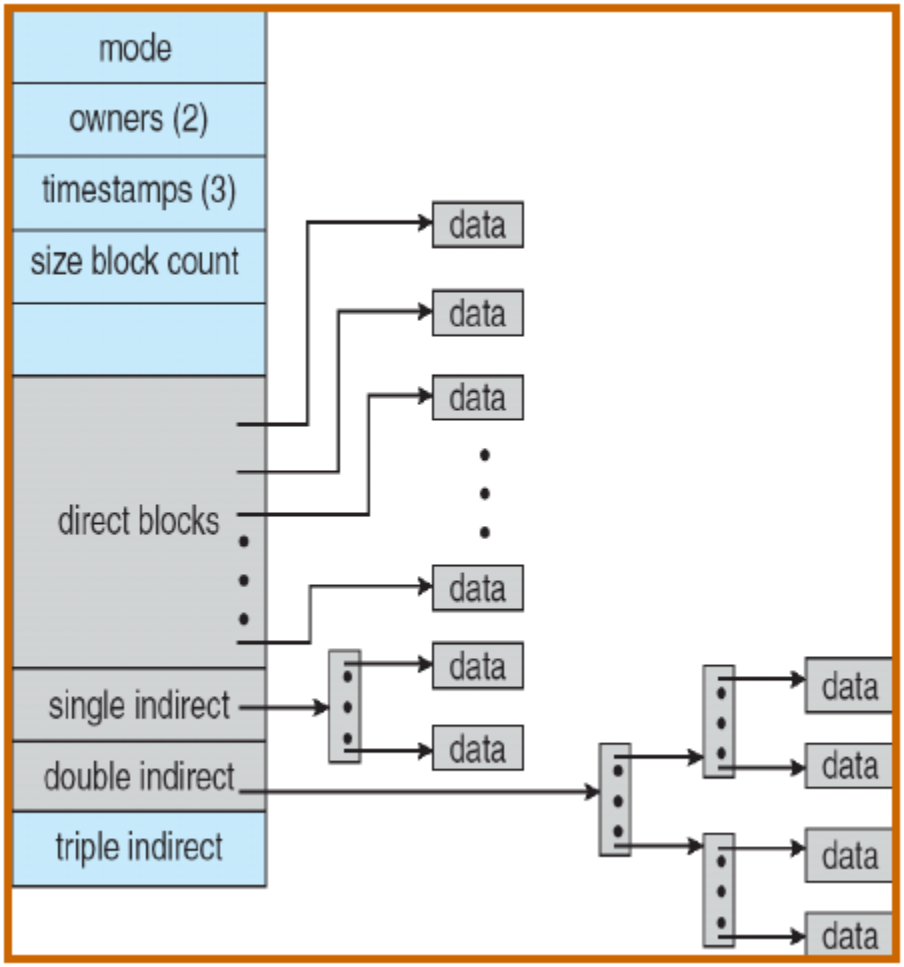
Asignación indexada (*Indexed Allocation*)

- Los bloques son accedidos directamente a través del bloque de indexación (*index block*).
- El bloque de indexación ocupa lugar. Se trata de que sea lo más pequeño posible, pero limita la cantidad de bloques.
- Una posible alternativa es indexación en varios niveles. Algunos índices hacen referencia a bloques directos y otros a bloques de indexación.



Asignación indexada (*Indexed Allocation*)

- En UNIX los bloques de control de archivos tienen bloques de indexación directa, de uno, dos y hasta de tres niveles de indexación.
- Esto permite representar archivos muy grandes.



Administración del espacio libre

- En el sistema de archivos es necesario mantener que bloques están ocupados y cuales están libres. Alternativas posibles para la administración de los bloques:
  - **Vector de bits** (*Bit Vector, Bit Map*): se dispone de un bit para cada bloque de datos del sistema, que representa si está ocupado o libre.
  - **Lista de bloques libres** (*Linked list*): Se mantiene una lista encadenada con los bloques libres a través de los bloques. Es necesario una referencia al primer bloque.
  - **Agrupación** (*Grouping*): es una variación de la lista encadenada. En cada bloque de la lista se contiene un grupo de bloques libres.
  - **Conteo** (*Counting*): se mantiene una lista en donde cada bloque contiene información de cuantos bloques contiguos, a partir de él, están libres.

Ejemplo UNIX

- Cada partición contiene un bloque descriptor del sistema de archivo llamado *super-block*.
- El *super-block* contiene:
  - Nombre del volúmen.
  - Cantidad máxima de archivos (inodos). Cantidad de archivos utilizados y libres.
  - Cantidad de bloques de datos, cantidad de bloques utilizados y libres.
  - Referencia a comienzo de bloques de datos, de indexación y de vector de bits.
  - Información de conteo.
  - etc.

Ejemplo UNIX

- La administración del espacio libre se realiza a través de mapa de bits (*bit vector*). Se disponen varios bloques al comienzo de la partición.
- El bloque de control de archivo es a la estructura *inode*. Los inodos son identificados por un número, que es único a nivel del sistema de archivos. Los inodos poseen un tipo:
  - archivo común
  - directorio
  - enlace simbólico
  - *pipes*
  - *socket*
- Utiliza un método de asignación por indexación.

Ejemplo UNIX

- Los directorios son representados como un archivo (inodo), en donde los datos son entradas que tienen los nombres de los archivos y el número de inodo correspondiente.
- Si es un *soft link*, se tiene la ruta (*path*) del archivo al cual referencian. Los *hard links* son tratados en forma natural, ya que la pertenencia de un archivo a un directorio está en los datos del directorio y se referencia al número de inodo.