Unidad 6: SISTEMAS DE ARCHIVOS

Introducción. Tipos. Funciones. Operaciones con y en los archivos. Jerarquías de datos. Tipos de registros. Métodos de control de acceso a la información. Administración de archivos. Directorio de un dispositivo-. Matriz de control de acceso. Estructura de directorios.

ADMINISTRADOR DE LA INFORMACIÓN

INTRODUCCIÓN

Todo sistema operativo debe poseer un módulo que se encargue del manejo de aspectos tales como el almacenamiento y recuperación de la información. Este módulo es el **Administrador de Información** que pertenece al **Administrador de Recursos. (**Comúnmente al conjunto de módulos del Administrador de Información se lo referencia como el **Sistema de Archivos** o **File System**).

FUNCIONES BÁSICAS DEL ADMINISTRADOR DE LA INFORMACIÓN

Llevar rastro de toda la información en el sistema a través de varias tablas

Estas tablas contienen para cada archivo:

- el nombre,
- ubicación,
- longitud (cantidad de registros del archivo),
- longitud del registro lógico,
- longitud del registro físico,
- formato de registros (fijo, variable, etc.),
- organización (secuencial, indexada, al azar, etc.),
- fecha de creación,
- fecha de expiración,
- derechos de acceso,
- extensiones.

Las extensiones de un archivo obedecen al hecho de que a menudo no es posible almacenar el mismo en un fragmento contiguo del periférico. Por lo tanto en el campo de extensiones se indican la cantidad de fragmentos que existen de ese archivo y se inicia a partir de la primera extensión (usualmente denominada alocación primaria del archivo) una cadena que apunta a las otras entradas del Directorio en donde continúan los otros pedazos del archivo. En adecuación a esto el tamaño total del archivo es el de su alocación primaria más el tamaño de todas sus extensiones.

Decidir que política es utilizada para determinar dónde y cómo la información es almacenada.

Algunos factores que influyen en esta política son:

- utilización efectiva del almacenamiento secundario,
- acceso eficiente,
- flexibilidad a usuarios
- protección de derechos de acceso sobre información pedida.

Asignación del recurso de la información

Una vez que la decisión de permitir a un proceso tener acceso a cierta información es efectuada, los módulos de ubicación de la información deben encontrar dicha información, hacer accesible dicha información al proceso y por último establecer los derechos de acceso apropiados.

Desasignación del recurso de la información

Una vez que la información no es necesitada más, las entradas en tablas asociadas a esta información pueden ser eliminadas. Si el usuario ha actualizado la información, la copia original de la información puede ser actualizada para posible uso de otros procesos.

MODELO GENERAL DE UN SISTEMA DE ARCHIVOS

Para el Sistema de Archivos que tomaremos como ejemplo, consideraremos que todos los módulos de la implementación se encuentran dispuestos en niveles o capas, de tal forma que dado un módulo, este sólo depende de aquellos módulos que se encuentran en niveles inferiores a él y sólo efectúa llamadas hacia estos módulos.

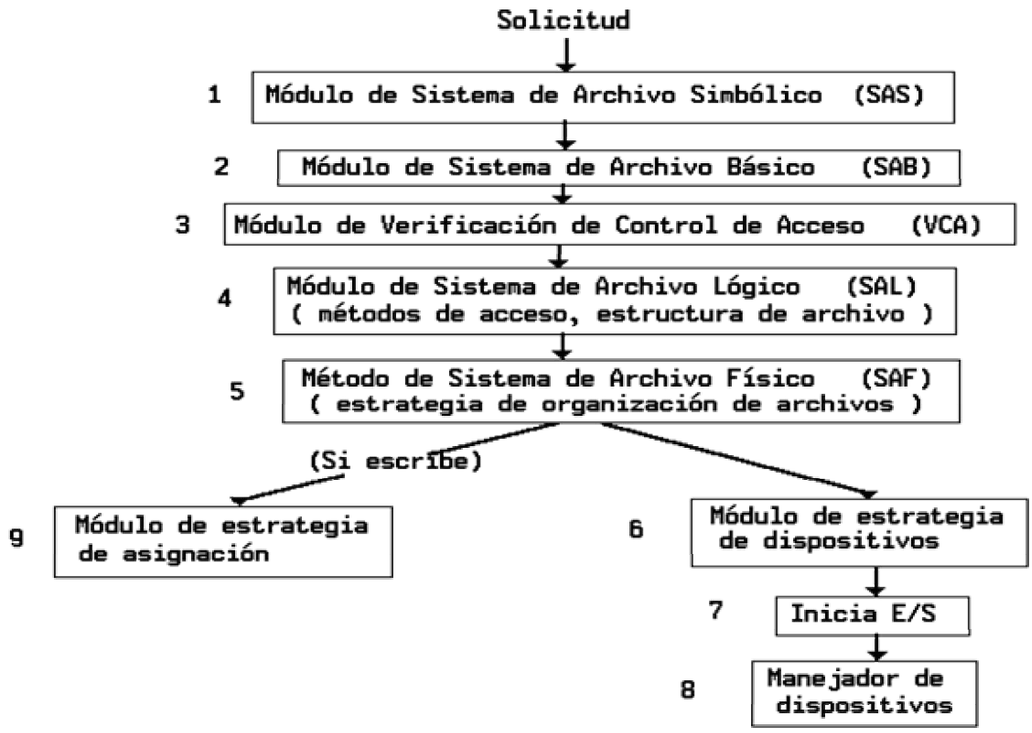

Fig 1: Modelo jerárquico de un Sistema de Archivos

También es necesario destacar que aunque los detalles específicos dados a continuación pueden variar significativamente de un sistema a otro, la estructura básica es común a la mayoría de los sistemas actuales. Es decir, dependiendo del sistema específico que tratemos, algunos de estos módulos se fusionan, se desglosan en aún más módulos, o incluso algunos desaparecen. Aún así la estructura subyacente sigue siendo la misma.

ESTRUCTURA Y MANTENIMIENTO DEL DIRECTORIO DE ARCHIVOS

Antes de describir el funcionamiento de los distintos módulos que conforman el Sistema de Archivos es necesario sentar algunas bases acerca de los siguientes términos:

Estructura del Directorio de Archivos:

Nosotros visualizaremos al Directorio de Archivos como una tabla donde cada entrada en ella corresponde a un archivo. Definiendo a un **archivo** como una **colección de unidades de información relacionadas** llamadas **registros**.

Así, el contenido de una entrada del Directorio de Archivos del ejemplo contendrá los siguientes datos:

- **Número de entrada**: se mantiene dicho número pues este se convertirá en el identificador del archivo dentro del sistema.
- Longitud de registro lógico,
- Cantidad de registros lógicos,
- Longitud del registro físico,
- Formato de los registros,
- Organización,
- Dirección al primer bloque físico,
- **Protección y control de acceso**: indica qué usuario tiene derecho de acceso sobre el archivo y que tipo de operaciones se pueden realizar sobre el archivo (solo lectura, lectura/escritura, etc.).

Desde el **2º al 7º** elemento de la tabla sirven para poder ubicar y mapear un registro lógico en su verdadera dirección física en el almacenamiento secundario.

Creación de una entrada del Directorio de Archivos:

Los comandos **CREAR** y **BORRAR** incorporan o eliminan respectivamente una entrada en el Directorio de Archivos. Así, una posible sintaxis del comando **CREAR** sería:

CREAR nombre-archivo, longitud-registro-lógico, cantidad-registros-lógicos, [ubicación- primer-bloque-físico], [formato], [organización]

Ubicación del Directorio de Archivos:

Si todo el Directorio de Archivos fuera mantenido todo el tiempo en memoria principal, se necesitaría una gran cantidad de memoria sólo para este propósito, por lo que surge la idea de mantener al Directorio de Archivos como un archivo dentro de un volumen de almacenamiento (cinta, diskette, disco, etc.).

Si bien se soluciona el problema del desperdicio de memoria principal, aparece el inconveniente de que la búsqueda de una entrada en dicho directorio puede ser considerablemente larga, si este constara, por ejemplo, de unas cuantas miles de entradas.

Este problema se soluciona manteniendo en memoria aquellas entradas del Directorio que pertenecen a archivos que fueron referenciados anteriormente. En llamadas subsiguientes no habrá desperdicio de tiempo de E/S.

Muchos sistemas operativos constan de pedidos especiales tales como **ABRIR** y **CERRAR** (**Open** y **Close** respectivamente) un archivo determinado. **ABRIR** coloca en memoria la entrada en la tabla perteneciente al archivo en cuestión y **CERRAR** constituye su contrapartida.

SISTEMA DE ARCHIVO SIMBOLICO

El primer módulo que es llamado cuando hacemos un pedido al Sistema de Archivos es el módulo denominado **Sistema de Archivo Simbólico (SAS).**

Una típica llamada sería por ejemplo: CALL SAS(READ,"JUAN",4,1200)

Donde estamos pidiendo que se lea el registro lógico número **4** del archivo "**JUAN**", para colocar su contenido en la dirección **1200** de memoria principal.

El **SAS** usa el nombre del archivo para localizar la única entrada que posee el archivo en el **Directorio de Archivos**.

Si existiera una correspondencia biunívoca entre el nombre que puede tener un archivo y el archivo en cuestión, el sistema adolecería de una gran falla. Pues no se podría referenciar a un archivo por diferentes nombres y distintos archivos no podrían tener el mismo nombre.

Para implementar tal facilidad se divide el Directorio de Archivos en dos partes. Un **Directorio de Archivo Simbólico (DAS)** y un **Directorio de Archivo Básico (DAB)** como se muestra en la figura.

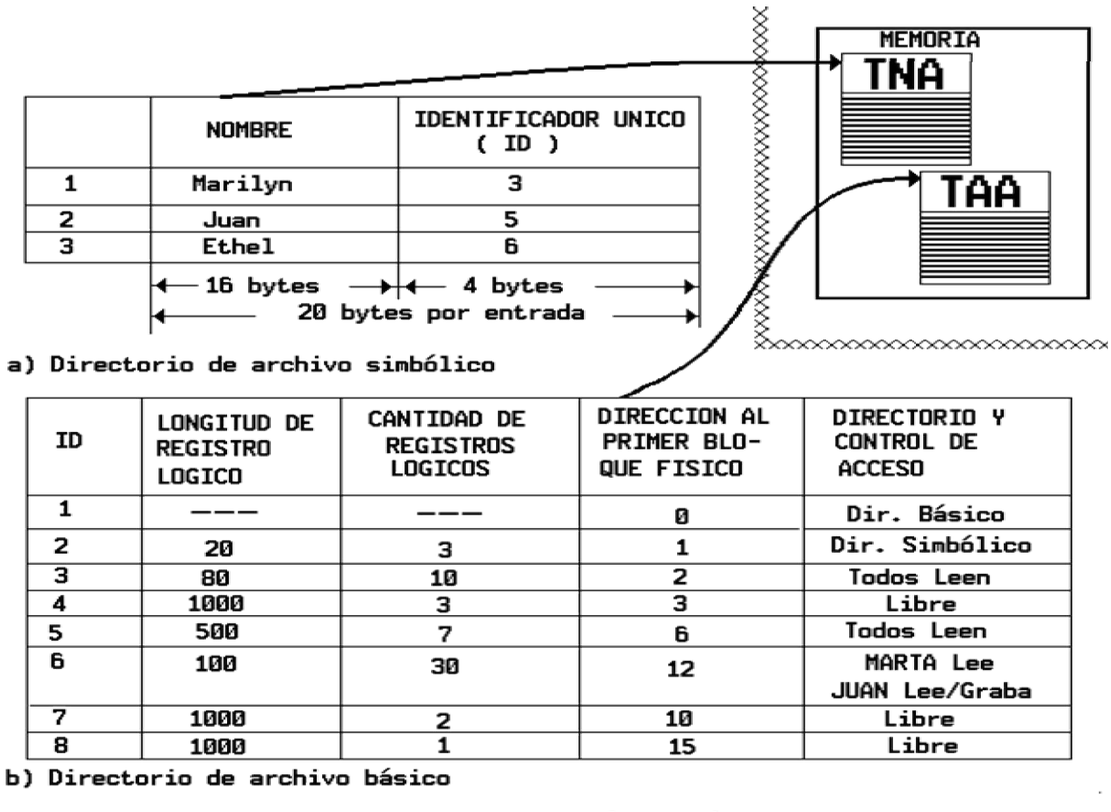

Fig. 2: Directorios Simbólico y Básico

El **Sistema de Archivo Simbólico** debe buscar en el **Directorio de Archivo Simbólico** la entrada perteneciente al archivo requerido y de esta forma encontrar su único **identificador (ID)** dentro del sistema, para pasárselo al módulo denominado **Sistema de Archivo Básico (SAB).**

En nuestro ejemplo **CALL SAS(READ,"JUAN",4,1200)** devolvería **5** donde **5** es el **ID** de del archivo "**JUAN**".

Normalmente como el **DAS** es mantenido en el periférico de almacenamiento, lo que se hace es copiar en memoria las **entradas del DAS** que corresponden a **archivos en uso** (llamados archivos abiertos o activos), de tal forma que estas entradas son usadas para evitar E/S sobre la misma zona en el periférico. Esta tabla que se mantiene en memoria principal con las entradas del **DAS** correspondientes a archivos abiertos recibe el nombre de **Tabla de Nombres Activos (TNA)**

SISTEMA DE ARCHIVO BASICO

El segundo módulo que es llamado en la secuencia se denomina **Sistema de Archivo Básico (SAB)**.

La llamada sería de esta forma: **CALL SAB(READ,5,4,1200)**

Donde todos los parámetros son iguales a la llamada del módulo **SAS**, a **excepción del segundo** parámetro que constituye el **identificador** que le pasó el **SAS**.

El **SAB** se vale del identificador del archivo (**ID**) para localizar la entrada correspondiente a este en el **Directorio de Archivo Básico**.

Dicha entrada es mantenida en memoria principal con el objeto de ahorrar posteriores E/S buscando la misma entrada.

La tabla que alberga todas las entradas del Directorio de Archivo Básico de los archivos abiertos recibe el nombre de **Tabla de Archivos Activos (TAA)**.

Esta tabla se genera y mantiene en memoria principal a diferencia del directorio básico (DAB) y del directorio simbólico (DAS). Su entrada consta de la información que puede visualizarse en la figura siguiente.

Fig. 3: Tabla de Archivos Activos

Para la próxima etapa -**Verificación de Control de Acceso (VCA)**- utilizaremos en vez del **ID** del archivo, su entrada correspondiente en la TAA, la cual contiene la información del archivo **ID 5**.

La invocación al módulo de Verificación de Control de Acceso es de la siguiente forma: **CALL VCA(READ,2,4,1200)**

Donde **2** es la entrada **2** de la **TAA** que corresponde al archivo "**JUAN**" y los demás son exactamente los mismos parámetros de la llamada al módulo anterior.

En resumen podemos decir que el módulo **SAB** se encarga de:

1. El Directorio de Archivo Básico.
2. La Tabla de Archivos Activos.
3. La comunicación con el módulo VCA.

VERIFICACION DE CONTROL DE ACCESO

Este módulo **actúa como un punto de control** entre el **Sistema de Archivo Básico** y el **módulo de Sistema de Archivo Lógico**, de tal forma que verifica si la función que se quiere realizar sobre el archivo en cuestión (**READ**, **WRITE**, etc.) está explicitada en la entrada correspondiente al archivo en la **Tabla de Archivos Activos**. En caso que no sea permitido realizar dicha operación sobre el archivo estamos en presencia de una condición de error, por lo cual el pedido al Sistema de Archivos es abortado. Si efectivamente se puede realizar esa operación entonces el control pasa directamente al módulo de **Sistema de Archivo Lógico**.

SISTEMA DE ARCHIVO LOGICO

Una llamada al módulo de **Sistema de Archivo Lógico (SAL)** posee la misma sintaxis que la llamada al módulo anterior. Luego, para el ejemplo que damos quedaría: **CALL SAL(READ,2,4,1200)**

El Sistema de Archivo Lógico convierte el pedido de un registro lógico en el pedido de una secuencia de bytes lógicos, la cual se entrega al **Sistema de Archivo Físico (SAF).**

Esto es así, puesto que para el **SAF** un archivo no es más que una secuencia de bytes sin ningún tipo de formato.

Para el ejemplo, vamos a suponer un formato de registro lógico de longitud fija, luego la conversión necesaria la podemos obtener de la información de la entrada en la **TAA**.

Dirección byte lógico (dbl) = (Nro. de registro - 1) * longitud del registro lógico = 3 * 500 = 1500

long. de la secuencia de bytes lógicos (lsb) = long. del registro lógico

Teniendo ya calculado el comienzo de la secuencia de bytes y su longitud, el **SAL** invoca al módulo de **Sistema de Archivo Físico (SAF): CALL SAF(READ,2,1500,500,1200) 1500 es dbl y 500 el lsb**

Donde el tercer y cuarto parámetro corresponden al dbl y el lsb respectivamente y el resto de parámetros son exactamente los mismos que los de la llamada el módulo anterior.

SISTEMA DE ARCHIVO FISICO

El **SAF** tiene por función determinar en qué **bloque físico** del dispositivo de almacenamiento se encuentra la secuencia de bytes lógicos pasados por el **SAL**, para lo cual se vale de la entrada en la **TAA** más el **dbl** y el **lsb**.

El **bloque** es entonces **leído y colocado en un buffer** preasignado en memoria principal, luego es extraído a partir de este buffer la secuencia de bytes pedidos y colocados en el área de buffer del usuario.

Para realizar estos cálculos el **SAF** debe saber las funciones de mapeo y longitud del bloque físico que utiliza cada periférico de almacenamiento. Si estas fueran las mismas para todo tipo de periférico podrían estar tranquilamente insertas en las mismas rutinas del **SAF**, pero tal cosa no sucede pues hay variaciones de un periférico a otro.

Esto se resuelve manteniendo tal información en el mismo volumen de almacenamiento, de tal forma que cuando se inicializa el sistema toda esta información pueda ser leída en una entrada de la **Tabla de Archivos Activos.**

Si se hace un pedido de escritura y el bloque sobre el cual se quiere escribir no está asignado previamente entonces el **Sistema de Archivo Físico** invoca al **Módulo de Estrategia de Asignación (MEA)** para que este le proporcione un bloque libre en memoria secundaria sobre el cual pueda efectuar la escritura.

MODULO DE ESTRATEGIA DE ASIGNACION

Este módulo se encarga de **llevar un registro del espacio libre disponible en el almacenamiento secundario**, tal información aparecerá reflejada en la **TAA**.

En la **figura 2b** se muestra como se puede hacer uso del Directorio de Archivo Básico para llevar cuenta de esta información, es decir tratamos a los espacios libres en memoria secundaria como si fueran archivos. Obviamente existen otras técnicas mejores para tratar este problema.

MODULO DE ESTRATEGIA DE PERIFERICO

Este módulo convierte el número de bloque físico en el formato adecuado para el periférico en cuestión (por ejemplo bloque 7 = pista 3 ,sector 23). Además de esto, inicializa los comandos adecuados de E/S para el tipo de periférico sobre el cual se va a realizar la operación.

Cabe destacar que todos los módulos anteriormente citados son totalmente independientes de cualquier tipo de periférico con excepción del último nombrado. De aquí en más, el control pasa a manos del **Administrador de Entrada-Salida.**

Resumen de los módulos

A modo de resumen se reseña brevemente todos los módulos antes mencionados:

1) Sistema de archivos simbólicos (SAS):

Transforma el nombre del archivo en el identifi- cador único del Directorio de archivos. Utiliza la Tabla de nombres activos (TNA) y el directo- rio de archivos simbólico (DAS).

2) Sistema de archivos básico (SAB):

Copia la entrada de la VTOC en memoria. Utiliza el directorio de archivos básicos (DAB) y la Tabla de archivos activos (TAA).

3) Verificación de control de acceso (VCA):

Verifica los permisos de acceso al archivo.

4) Sistema de archivo lógico (SAL):

Transforma el pedido lógico en un hilo de bytes lógicos.

5) Sistema de archivo físico (SAF):

Calcula la dirección física.

6) Módulo de estrategia de asignación (MEA):

Consigue espacio disponible en el periférico (casos de grabación).

7) Módulo de estrategia de periférico:

Transforma la dirección física según las características exactas del periférico requerido.

ESTRUCTURAS DE DIRECTORIOS

Al hablar del **Sistema de Archivos Básico** hemos aclarado que el mismo opera sobre el **Directorio de Archivos Básico (DAB)**. Usualmente este directorio se encuentra almacenado el algún periférico del sistema que posea velocidad alta.

Si pensamos un momento la cantidad de archivos que pueden llegar a existir en cualquier centro de cómputos (aún siendo pequeño) comprenderemos inmediatamente que contar con un único directorio básico en forma de una interminable tabla que deberá ser accedida toda vez que se requiera un archivo, resulta poco práctico.

Por este motivo es conveniente estructurar los directorios en forma jerárquica para proveer una mejor forma de acceso, un marco de trabajo ordenado y protección para todos los usuarios en su conjunto.

DIRECTORIOS DE UN SOLO NIVEL

Una de las formas de estructurar un directorio puede verse en la **Fig. 4**. Nótese que en este caso la entrada en el directorio de archivos básico está apuntando a una estructura que mapea todos los bloques del archivo real en disco.

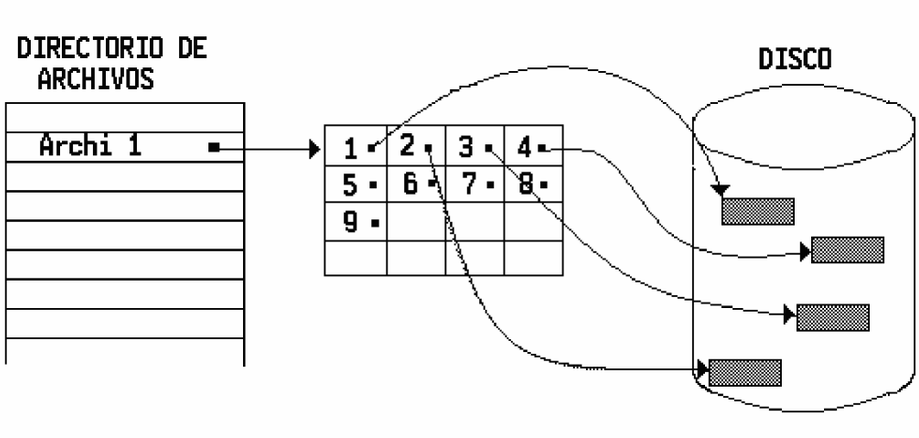

Fig. 4: Estructura de directorio de un solo nivel

La cantidad de accesos necesarios para llegar a un archivo en una estructura de este tipo serán **3**, uno para llegar al directorio básico, uno para llegar a la tabla de mapeo y uno para llegar al archivo real.

La tabla de mapeo se utiliza para optimizar la utilización del espacio en el disco aunque cuenta con el inconveniente de que la información puede estar muy dispersa dentro del dispositivo degradando todo tipo de proceso que requiera leer una gran cantidad de registros de este archivo.

DIRECTORIOS DE VARIOS NIVELES

Una estructura más compleja de directorios puede visualizarse en la **Fig. 5**. En este caso la cantidad de accesos para llegar a un archivo dependerá de la profundidad de niveles del directorio. En forma genérica podemos decir que para llegar a una entrada en el nivel **n** se requerirán no menos de **n+1** accesos a disco, debido a que es necesario siempre recorrer los **n niveles** que apuntan a dicho archivo para finalmente acceder finalmente al archivo en si.

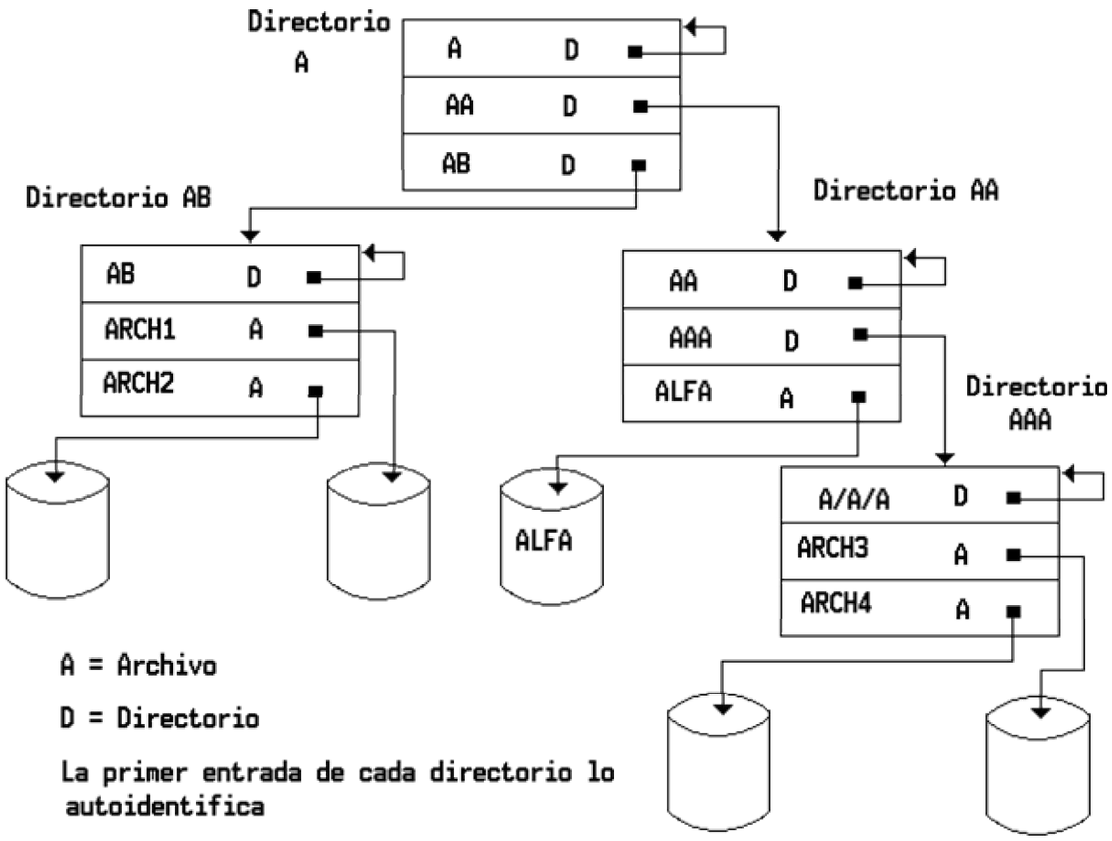

Fig. 5: Estructura de directorios más compleja

Nótese en el ejemplo que para llegar al archivo **ALFA** se deberá ingresar al sistema por el directorio **A**, de allí pasar al segundo nivel en el directorio **AA** y de allí acceder al archivo; por lo tanto para proveer el camino completo para referenciarse al archivo un usuario debería especificar el nombre del archivo de la forma **A/AA/ALFA** pudiendo utilizarse como separadores la '/', el '.', o cualquier otro similar provisto por el lenguaje de control.

En las estructuras de este tipo debe tenerse presente que en las entradas de los directorios debe existir algún campo que permita diferenciar si la información contenida en esa entrada corresponde a un directorio o a un archivo (por ejemplo la entrada correspondiente al directorio **AB** y la entrada correspondiente al archivo **ALFA** en el directorio **AA** segundo nivel).

ESTRUCTURAS DE CONTROL DE ACCESO

El control de acceso a un archivo está dado por una serie de permisos que son controlados por el módulo de **Verificación de Control de Acceso (VCA).**

Tales permisos se almacenan conjuntamente con la información del **directorio básico** o alguno de sus niveles y pueden constituir asimismo archivos de permisos. Estos datos serán copiados por el **Sistema de archivos Básicos** en la **Tabla de Archivos Activos** y luego consultados por el **VCA** para autorizar el acceso.

Una vez liberado el archivo y de haber ocurrido algún cambio en los permisos la información deberá ser devuelta a su lugar de almacenamiento en memoria secundaria.

LISTA DE CONTROL DE ACCESO

El concepto de la **Lista de Control de Accesos (LCA)** se basa en que **para cada archivo** en el sistema **se asocia una lista de permisos** (que puede ser otro archivo) en donde se especifica para cada usuario que tipo de acciones puede realizar sobre ese archivo.

En el ejemplo de la **Fig. 6** el usuario **JOSE** puede **Leer** sobre el archivo **ALFA** y el usuario **MARIA** puede **Leer/Grabar** sobre el archivo **ALFA**.

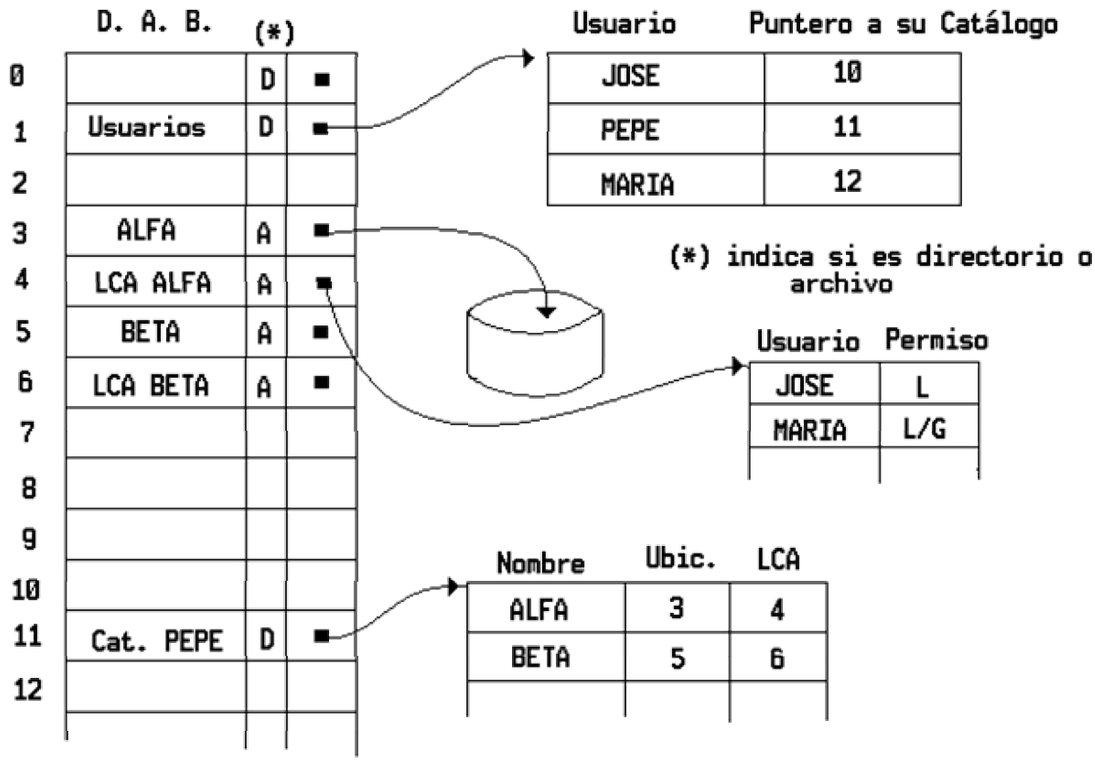

Fig. 6: Lista de Control de Acceso

Suele existir también el permiso de **Owner** que indica el nombre del usuario que originalmente creó ese archivo y que puede realizar todo tipo de acciones sobre él. En el ejemplo no se ha especificado este permiso debido a que para llegar al archivo **ALFA** se debe ingresar obligatoriamente por el directorio del usuario **PEPE** asumiéndose entonces que dicho usuario es el **Owner**.

Pueden existir también **permisos de Ejecución** cuando se trata de archivos que corresponden a códigos objeto de programas. Este permiso es muy útil ya que solamente permite el acceso a tal archivo si la acción deseada es Ejecutar y en cambio lo niega si la acción deseada es de Lectura (**previene copias ilegales**). La acción se puede diferenciar en virtud de la naturaleza del proceso que la invoca (un **COPY** es diferente de un **EXEC**).

Nótese que por la estructura que poseen el directorio de ejemplo es requisito indispensable que para que un usuario llegue a un archivo que no le es propio conozca el nombre del usuario que es su propietario a fin de proveer la información de la vía necesaria para ubicar ese archivo (**path**). En tal estructura un acceso al directorio básico con el nombre **ALFA** no sería útil debido a que el módulo **VCA** exige la verificación de la **LCA** del archivo y tal información solamente existe a nivel del directorio del usuario **PEPE** (obvio, el acceso por defecto es Ninguno).

La forma más sencilla de proteger el directorio básico es aprovechar la entrada cero, que existe siempre en todo tipo de directorio y que sirve para autoidentificarse, para agregar allí la información de donde se encuentra su LCA.

En **LCA** la capacidad de acceso a un archivo pertenece al mismo archivo.

LISTA DE CONTROL DE USUARIO

A diferencia de la LCA la **Lista de Control de Usuario (LCU)** se basa en el concepto de que para cada usuario existente en el sistema hay una lista de los archivos a los cuales puede acceder y que acciones puede realizar sobre ellos. En la **Fig. 7** podemos ver como con prácticamente la misma estructura de directorios anterior construimos una LCU.

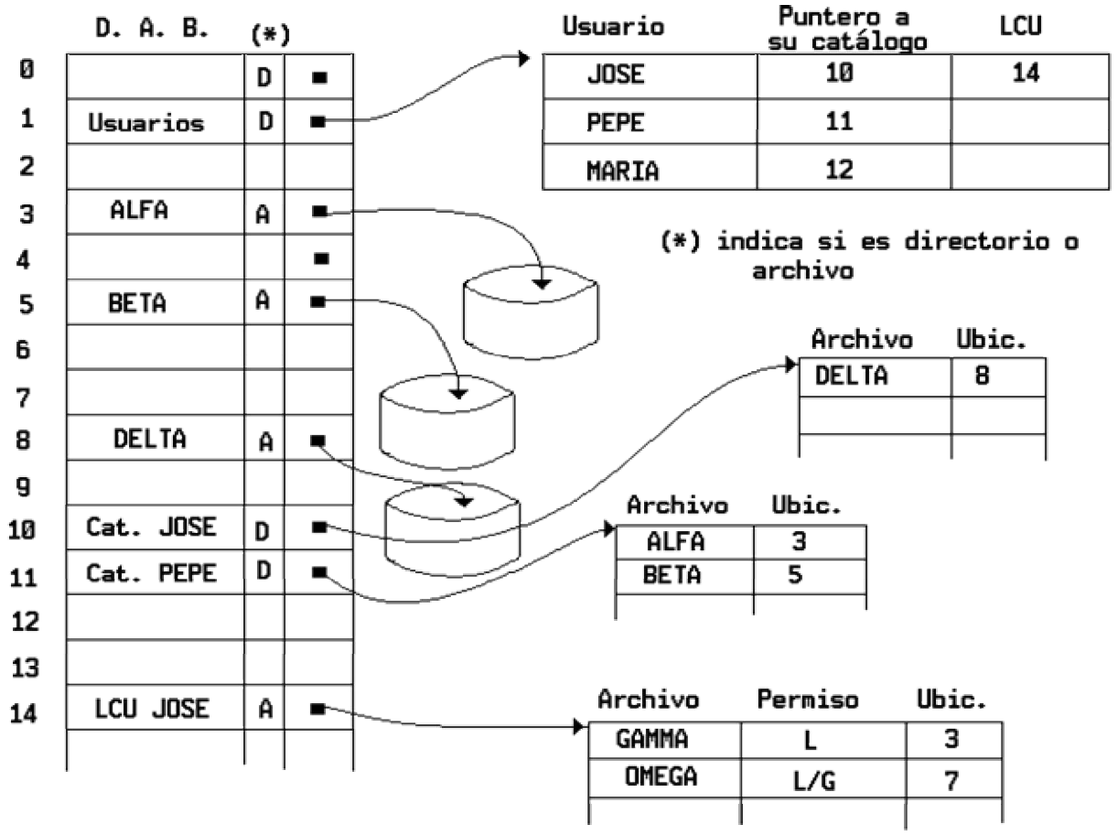

Fig. 7: Sistema de Lista de Control de Usuario (LCU)

Para cada usuario tenemos una entrada que corresponde al directorio de los archivos que le son propios y un puntero a una lista de archivos que indica la ubicación del archivo en el **DAB**, su nombre y el permiso de acceso.

Aquí puede verse inmediatamente como un mismo archivo puede ser visto con nombres diferentes por varios usuarios (JOSE/GAMMA y PEPE/ALFA son el mismo archivo).

En **LCU** la capacidad de acceso a un archivo pertenece al usuario que desea accederlo.

Comparación entre LCA y LCU

En la **LCA la baja** de un archivo **no presenta mayores inconvenientes** ya que se elimina la entrada del archivo en el catálogo del usuario Owner y luego se liberan las entradas correspondientes al archivo y a su LCA en el DAB.

En cambio en **LCU** presenta un inconveniente, ya que no se sabe a priori qué otros usuarios apuntan a ese archivo. Una solución es al eliminar el archivo **recorrer todas** las **LCU** de los otros usuarios rastreando aquellas cuyo puntero al básico coincida con el archivo eliminado para darlas de baja. Esta tarea se puede ver facilitada si se encadenan las entradas para un mismo archivo que correspondan a diferentes usuarios, pero por otra parte complica el mecanismo de mantenimiento y actualización de archivos.

La **LCU** por su naturaleza **permite fácilmente la existencia de diferentes nombres para un mismo archivo** lo que beneficia al sistema desde un punto de vista de seguridad y confidencialidad, hemos visto que esto no es tan sencillo en el sistema **LCA**.

No es fácil implementar un **permiso genérico** (del tipo Todos Leen) en una **LCU** ya que debería colocarse en la **LCU** de cada usuario la referencia al archivo deseado.

OPERACION DE OTRAS INSTRUCCIONES DE E/S.

Hemos visto en detalle las diferentes acciones que desencadena una instrucción de lectura (READ) en el sistema de administración de la información. A continuación detallaremos las acciones que se suceden con otras instrucciones típicas de E/S.

Instrucción OPEN

Para que un **archivo** pueda **abrirse**, éste deberá ser **asignado previamente**.

La asignación puede hacerse **dentro del programa** con la ejecución de la instrucción de apertura del archivo o **separadamente** (a nivel etapa).

La asignación propiamente dicha realiza los siguientes pasos:

- generar la entrada para el dispositivo apuntada desde el BCP.
- generar la entrada para el archivo apuntada también desde el BCP y asociar el nombre interno con el nombre externo.
- generar una entrada en la **Tabla de Nombres de Archivos (TNA)**.

En aquellos casos en que el lenguaje no provee una instrucción específica de apertura generalmente la primera instrucción de lectura o grabación del archivo es la que desencadena las acciones de la instrucción OPEN.

Apertura de archivo con asignación a nivel etapa (OPEN sin ASSIGN)

Como **ya se realizó la asignación**, ya **existe** una entrada en la **Tabla de Nombres Activos (TNA)**, con esta información se procede entonces a buscar el archivo en la **Tabla de Archivos Activos (TAA).**

- **Si se encuentra**, implica que el archivo estaba siendo usado por otro proceso y se controlan los permisos y concurrencia según los campos respectivos en la **TAA**, y luego se asocia el dispositivo y el archivo al proceso.
- **Si no se encuentra** el archivo en la **TAA** se deberá **generar una entrada** en la misma accediendo al **Directorio de Archivos Básicos (DAB)** en el disco y copiando la entrada correspondiente al archivo en memoria (**TAA**). Para la búsqueda del **DAB** es posible que haya que recorrer los volúmenes. Se controlan los permisos en el disco (que también se copian en la **TAA**) y finalmente se asocia el dispositivo y el archivo al proceso.

Es importante tener en cuenta que si el archivo es de escritura (**OUTPUT**) y se lo utiliza por primera vez se debe solicitar la asignación del espacio libre al **Módulo de Estrategia de Asignación (MEA).**

Apertura de archivo con asignación de dispositivos (OPEN con ASSIGN)

Se genera una entrada para el dispositivo y otra entrada para el archivo en el BCP. Como el **OPEN** debe realizar la asignación, asocia el nombre interno con el nombre externo generando una entrada para ese archivo en la **Tabla de Nombres de Archivos (TNA)**

A partir de este punto la apertura se realiza igual que en el apartado anterior.

Instrucción CLOSE

Se busca la entrada del archivo en la **Tabla de Nombres Activos (TNA)**. Con el identificador que se extrae de la **TNA**, se accede a la **TAA**.

Por el **Sistema de Archivos Físicos** se puede saber si hay algún bloque que deba ser grabado o no (solamente se chequea esto último si el archivo fue utilizado de output o de input-output). Si éste no fue grabado, se

debe realizar una E/S física (en este caso un **WRITE**), se llama al **Módulo de Estrategia de Asignación (MEA)**, luego al **Módulo de Estrategia de Periférico** y se lanza la E/S física.

Si el archivo se está usando concurrentemente con otros procesos se elimina el proceso de la lista de archivos asociados al proceso en la **TAA** y se decrementa en uno el campo concurrencia de la misma tabla.

Si el archivo no se usa concurrentemente, es decir el campo concurrencia es igual a 1, se verifica si su entrada en la **TAA** fue modificada (casos de output, input-output o información de fechas), por lo tanto se debe actualizar esta información en el disco. Con **Sistema de Archivos Básico (SAB)** se graba esta información de la **TAA** al **Directorio de Archivos Básicos** en el disco y por último se borra la entrada del archivo de la lista de procesos que lo están usando y se borra la entrada del archivo en la **TAA** (concurrencia = 0).

Instrucción DELETE

Para borrar un archivo se debe controlar si el usuario que desea borrar el archivo tiene autorización para hacerlo (**VCA** - Puede ser el dueño o tener permiso de borrado). Si no está autorizado, se produce un error y no se puede realizar la acción.

Caso contrario, si se trata de un esquema de protección de tipo **LCU**, se recorren las listas de los usuarios buscando aquellas que apuntan a la entrada del archivo en el Básico y se las elimina. Luego se elimina físicamente el archivo, devolviendo el espacio ocupado por el mismo como espacio disponible. La eliminción física del archivo implica la baja de la entrada del directorio básico que apunta a la dirección física real del archivo en cuestión.

El recorrido según las listas es el siguiente:

LCU

- Por cada usuario del sistema:
  - Ingresar en la **LCU** de los archivos que no le son propios.
  - Eliminar el archivo de la lista
- Para el usuario dueño del archivo :
  - Elimina el archivo de la lista de archivos propios.

LCA

- Ingresar a la LCA del archivo y la elimina.
- Devuelve también la entrada de la LCA como espacio libre.

En la mayoría de los sistemas operativos de hoy en día se verifica la concurrencia de la acción de borrado respecto de la utilización del archivo por algún usuario en el momento en que se desea borrar el mismo a efectos de prevenir el fin anormal de una tarea.

ALGUNAS CONSIDERACIONES MÁS

SISTEMAS DE ARCHIVO LÓGICO O MÉTODO DE ACCESO (DIRECCIÓN LÓGICA)

El "cálculo de la dirección lógica" depende directamente de la estructura de los registros (formatos), de la organización del archivo y de la forma en que se quiere acceder a la información.

Las formas de llegar a la información, dependiendo de sus estructuras, organización y acceso es lo que conforman los **Métodos de Acceso**.

Luego, los Métodos de Acceso serán distintos (operarán de distinta manera) dependiendo de:

- la forma en que se quiere acceder a la información,
- del formato de los registros y
- la organización del archivo. Veamos algunos ejemplos.

Archivo Secuencial, Formato Fijo

En este tipo de archivo los registros tienen igual longitud, como podemos ver en la **Fig. 8.**

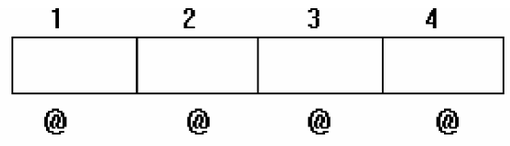

Fig. 8 Archivo Secuencial con Formato Fijo

Acceso Secuencial

En la **TAA** se mantiene un **NBI (Número de Byte Inicial) (dbli)** y un **NBF (Número de Byte Final) (dblf)**

NBI = inicialmente está en cero
NBF = número siguiente al del último registro grabado.

Luego, en este caso el (**SAL**) Método de Acceso actúa de la siguiente forma:

En lectura, luego de procesar cada registro se cambia **NBI = NBI + LR** (siendo LR la longitud del registro)

En escritura, si está permitido escribir luego del último registro (append), se cambia **NBF = NBF + LR**

Nótese que el **NBI** debe estar en la **TAA** y si el archivo es compartido debe existir además un **NBI** para cada proceso que lo comparte.

El **NBF** es un dato que está incluido en la **Tabla de Descripción del Archivo**, y debe incluirse en la **TAA** durante la operación de **OPEN**. Obviamente habrá uno solo.

En el momento de la creación de un archivo NBI y NBF mantienen el mismo valor.

Acceso Directo

Aquí se aplica el ejemplo ya visto, donde dado un NR (número de registro) se aplica: NBI = (NR - 1) * LR

Archivo Secuencial, Formato Variable.

En este tipo de archivos la longitud de cada registro es variable, de la siguiente forma:

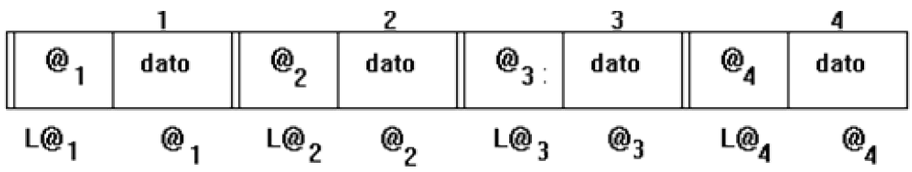

Fig. 9: Archivo Secuencial de Formato Variable

Donde

L@i = contiene la longitud del registro (sus longitudes son iguales)

@@i = contiene los datos en sí (sus longitudes son distintas)

Acceso Secuencial.

Aquí valen las mismas consideraciones realizadas antes para NBI y NBF. NBI = inicialmente en cero

luego el próximo registro se encontrará en:

$$NBI = NBI + L@n + @n$$

donde

L@n = longitud del campo longitud del registro n

@@n = longitud del dato

Hay que tener en cuenta que es necesario mantener en memoria un buffer de longitud suficiente para albergar el registro más largo posible de este archivo, dato que está en la descripción del mismo archivo.

Acceso Directo.

Si se quiere acceder al Registro Número 3, la forma de obtener su dirección es únicamente posible por medio de:

$$NBI = L@1 + @1 + L@2 +@ 2$$

Obviamente acceder a un registro de esta manera es muy ineficiente. Algunas maneras de mejorar esta situación serían:

1) Mantener el valor del último NBI de tal manera que si el próximo registro que se busca es superior al anterior buscado, se comienza desde ese.
2) Mantener el valor de todos los NBI leídos para usos futuros (Note/Point) (con la esperanza de reusarlos)
3) Mantener en memoria una tabla para todos los NBI (tiene la desventaja de una gran ocupación de espacio en memoria)

Para otras organizaciones será necesario que el Método de Acceso consulte las estructuras propias de esa organización.

Por ejemplo, en un Secuencial Indexado, dada su clave se deberá encontrar en los índices el valor del NBI; en un Random con clave, de deberá calcular a través de la clave y una función (si es provista) el valor del NBI.

A medida que las estructuras son más complejas y se brindan mayores facilidades de búsqueda (a través de relaciones, etc.) dentro del mismo archivo, e inclusive entre distintos archivos (referencias cruzadas, etc.) se encuentra que los Métodos de Acceso se transforman en Sistemas de Gestión de Bases de Datos.

Sistemas de Archivos Físico (SAF)

El cálculo de la dirección física se realiza de la siguiente manera:

Nº Bloque Relativo (NBR) = [DBL / Long. Bloque] y DF = NBR + Dir. Archivo (o dir. del 1º Bloque)

Además se ubica la posición de la información dentro del Bloque (byte dentro del bloque) como:

Posición Relativa Registro (PRR) = Resto [DBL / Longitud Bloque]

Si **DF** es una dirección cuyo contenido ya se encuentra dentro del buffer en memoria esto significa que **no es necesario** realizar una operación de E/S física, sólo es necesario trasladar la información al área del proceso (o pasar la dirección de la información), en caso contrario se sigue adelante con el resto de las funciones de la Administración de Archivos y las operaciones de E/S físicas.

Nótese aquí que si se estuviera trabajando con un **Sistema de Administración de Memoria Paginada por Demanda**, y el buffer no se encontrase en memoria se produciría una Interrupción por Falta de Página.

Pueden existir situaciones especiales como el caso de un registro lógico que supere el tamaño de los registros físicos, como por ejemplo:

CdB: Cabecera de bloque (usualmente 4 bytes de longitud, indica la longitud del bloque)

Fig. 10

Aquí habría que determinar cuantos bloques es necesario traer a memoria, que se calcula como:

Cantidad de bloques = [Long. Lógica / Long. Bloque] + 1

si **PRR = 0**, o sea el registro comienza al comienzo del bloque y

Cantidad de bloques = [Long. Lógica / Long. Bloque] + 2

si **PRR** ¹ 0, o sea que el registro comienza en el interior de un bloque.

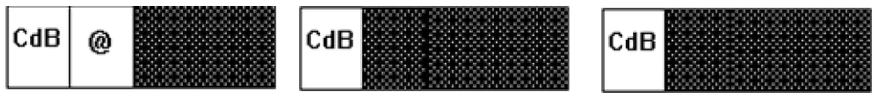

Fig. 11

Una vez que la información está en el buffer se trasladan @ bytes al área del usuario.