FILE SYSTEMS INTERFACE

FILE CONCEPT

- A collection of related bytes having meaning only to the creator. The file can be "free formed", indexed, structured, etc.

- The file is an entry in a directory.

- The file may have attributes (name, creator, date, type, permissions)

- The file may have structure (O.S. may or may not know about this.) It’s a tradeoff of power versus overhead. For example,

  a) An Operating System understands program image format in order to create a process.

  b) The UNIX shell understands what how directory files look. (In general the UNIX kernel doesn’t interpret files.)

  c) Usually the Operating System understands and interprets file types.

- Memory-mapped files are a special case.
  a) The Operating System "maps" or "connects" the data in the file with a region of memory in a process.
  b) When the file is closed, memory information is written back to the file.
  c) This is a great way to implement shared memory - have several processes open a memory-mapped file - they then each see the same information.

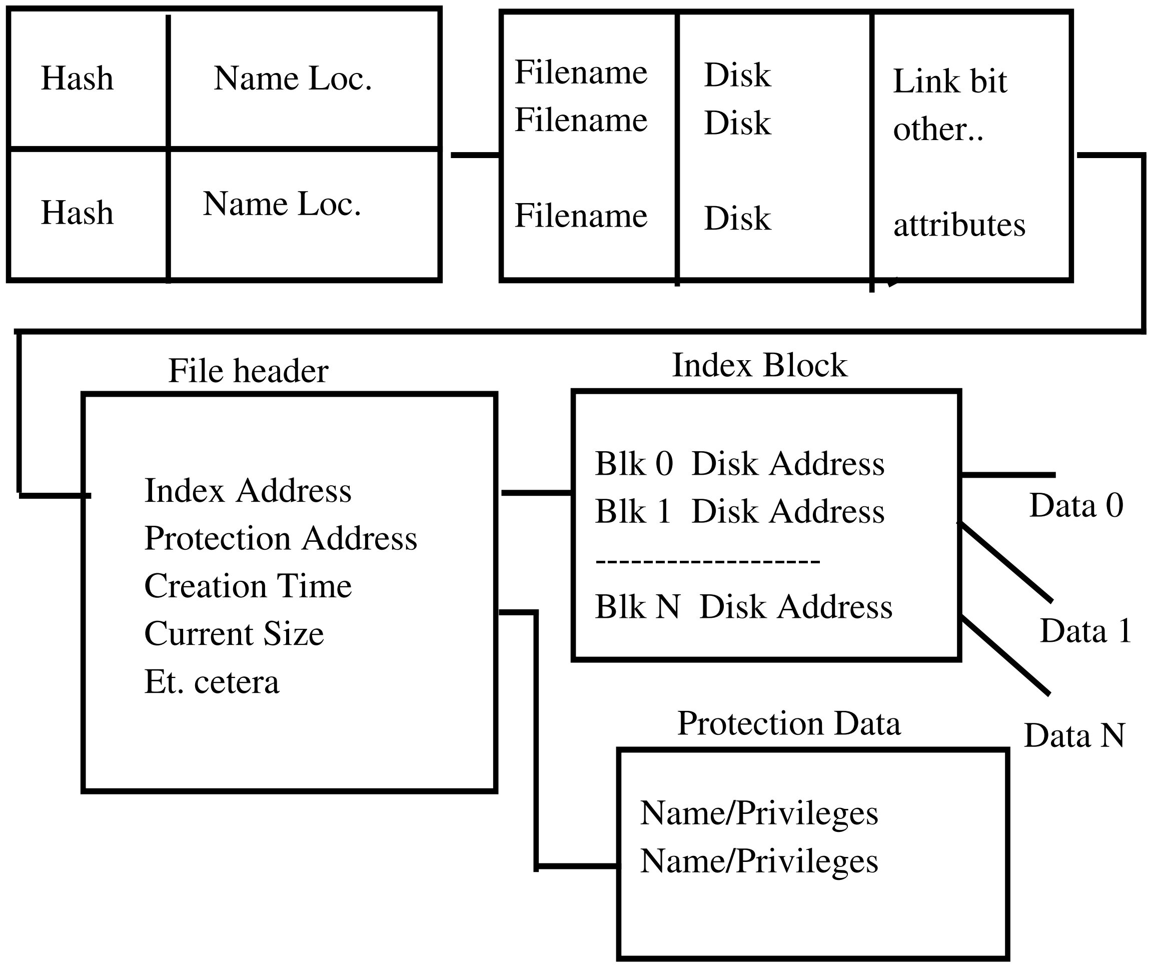
- Blocking occurs when some entity, (either the user or the Operating System) must pack bytes into a physical block.

  a) Block size is fixed for disks, variable for tape

  b) Size determines maximum internal fragmentation

  c) We can allow reference to a file as a set of logical records (addressable units) and then divide (or pack) logical records into physical blocks.

Example of File Structures



ACCESS METHODS

- If files had only one "chunk" of data, life would be simple. But for large files, the files themselves may contain structure, making access faster.
- As discussed in the last chapter, the file system may impose a structure on the way the file is allocated.

SEQUENTIAL ACCESS

- Implemented by the filesystem.
- Data is accessed one record right after the last.
- Reads cause a pointer to be moved ahead by one.
- Writes allocate space for the record and move the pointer to the new End Of File.
- Such a method is reasonable for tape

DIRECT ACCESS

- Method useful for disks.
- The file is viewed as a numbered sequence of blocks or records.
- There are no restrictions on which blocks are read/written in any order.
- User now says "read n" rather than "read next".
- "n" is a number relative to the beginning of file, not relative to an absolute physical disk location.

OTHER ACCESS METHODS

- Built on top of direct access and often implemented by a user utility.

Indexed ID plus pointer.

An index block says what's in each remaining block or contains pointers to blocks containing particular items. Suppose a file contains many blocks of data arranged by name alphabetically.

- **Example 1:** Index contains the name appearing as the first record in each block. There are as many index entries as there are blocks. **<<< FIGURE 10.5>>>**

- **Example 2:** Index contains the block number where "A" begins, where "B" begins, etc. Here there are only 26 index entries.

DIRECTORY STRUCTURE

- Directories maintain information about files:

- For a large number of files, may want a directory structure - directories under directories.

- Information maintained in a directory:

Name The user visible name.

Type The file is a directory, a program image, a user file, a link, etc.

Location Device and location on the device where the file header is located.

Size Number of bytes/words/blocks in the file.

Position Current next-read/next-write pointers.

Protection Access control on read/write/ execute/delete.

- Usage info Open count, time of creation/access, etc.

Mounting a filesystem occurs when the root of one filesystem is "grafted" into the existing tree of another filesystem.

- There is a need to **PROTECT** files and directories.
  Actions that might be protected include: **read, write, execute, append, delete, list**