Lecture 5 File Systems

Last Day: Space Management
- Free Pool

Today: Space Management
- Allocating Space

File Directories

See handout on file systems.

Review

- Disk blocks are either Free or allocated to a file.

- The set of free blocks is called the Free pool.

- The file system must

  Ⓐ Keep track of free blocks. } Last Day

  Ⓑ Allocate space for new files.

  Ⓒ Return blocks to free pool when users delete files.

(A) Managing the Free Pool.

Three main techniques:

① Linked list of free blocks.

② Linked list of segments

③ Bitmap

(B) Allocating Space for New Files:

Three main techniques:

① Segmented Allocation

② Linked Allocation.

③ Indexed Allocation.

① Segmented Allocation

- Free pool is a linked list of segments.

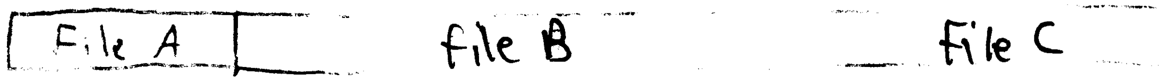
- When a file is created, the file system searches for a segment large enough to hold the entire file.

- Note. Each file is stored as one contiguous segment.

- Advantages:
    - Blocks of a file are close together (which minimizes seek time).
    - Very easy to find the i^{th} block of a file. (ie, can quickly find data in the middle of a file.)

Disa [illegible]

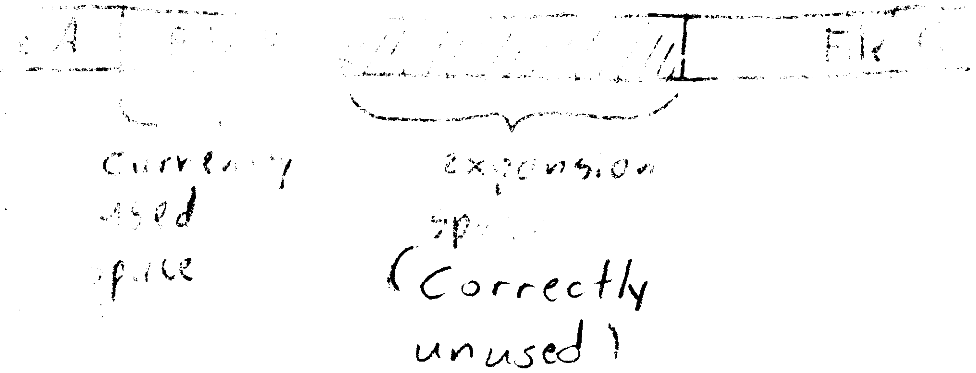
inflexible Can't increase file size



File B is hemmed in by Files A & C so it can't expand

Space is Wasted

To allow for expansion the file system must all be the maximum number of blocks that the file w ever need.

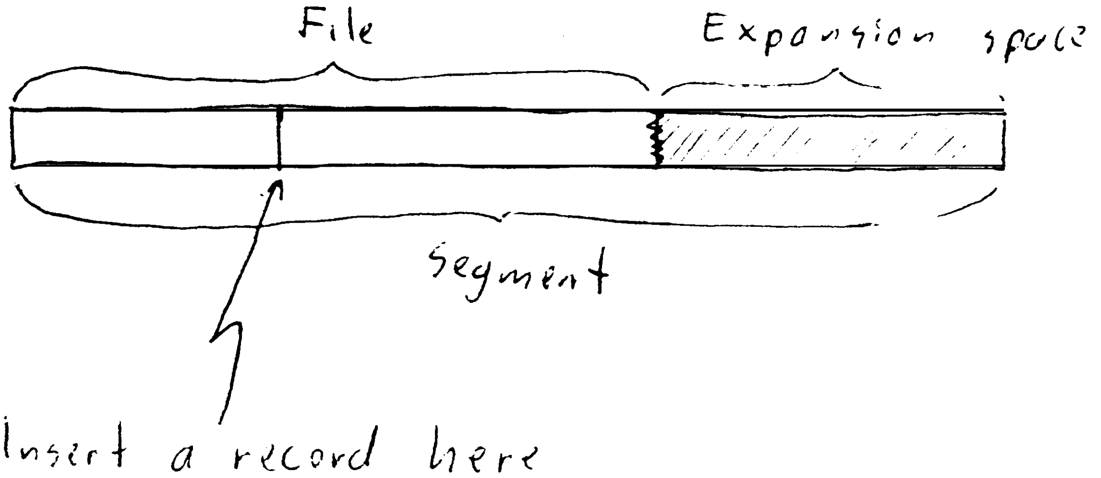


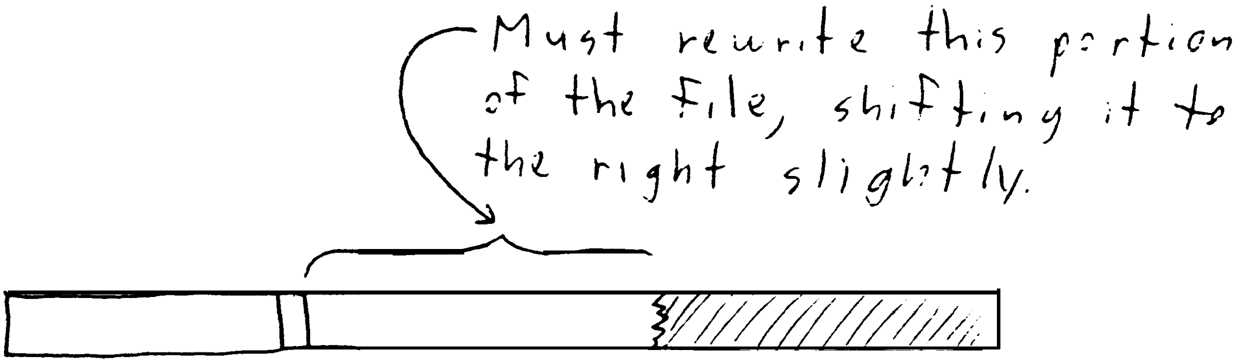
Disadvantages (Cont.)

Expensive to Expand:

To insert a record into the file, much of the File may have to be rewritten.

eg.




Must rewrite this portion of the file, shifting it to the right slightly.

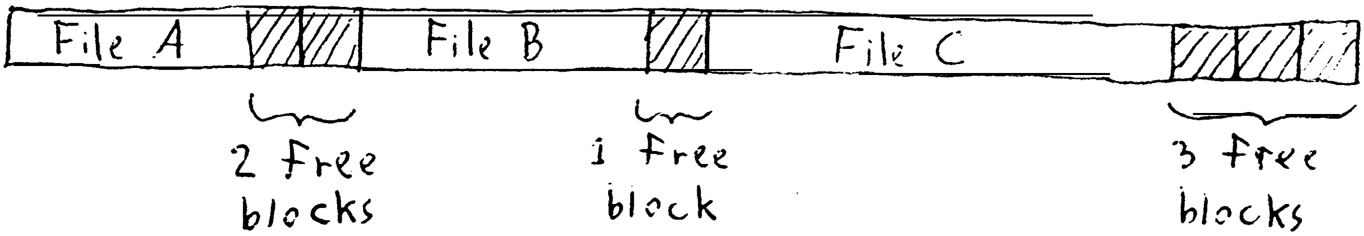
Disadvantages (cont.)

External Fragmentation:

There may be enough free blocks to store a file, but they may be scattered over the disk, so no contiguous segment is big enough to hold the file.

(∴ Can't store the file.)

eg.



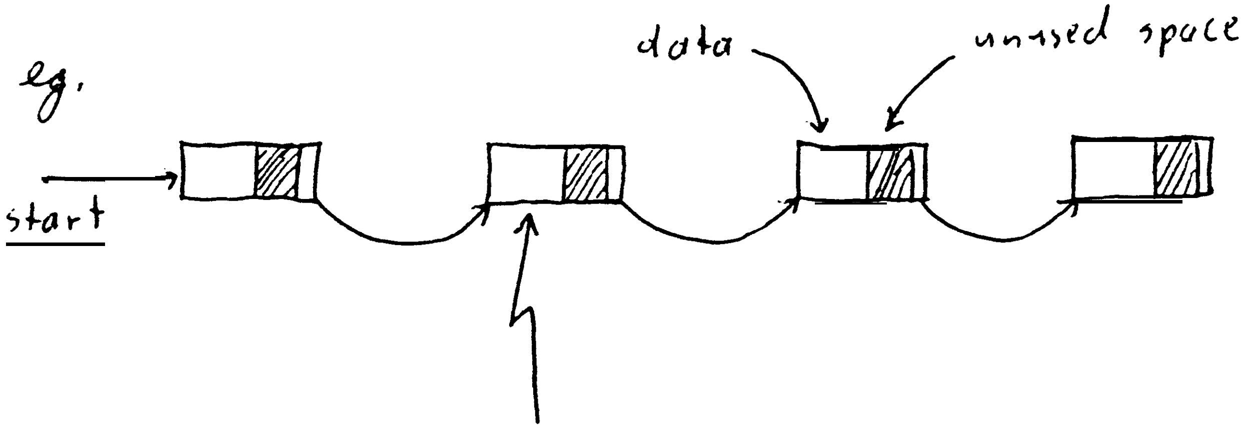
A new file (file D) needing 5 blocks cannot be stored, even though 6 blocks are free.

② Linked Allocation

- The blocks of a file form a linked list
- Note: The blocks of a file may now be scattered over the disk.

- Advantages:
  - No external fragmentation
  - File can easily expand or contract.
  - Can modify a file by writing only the affected blocks.

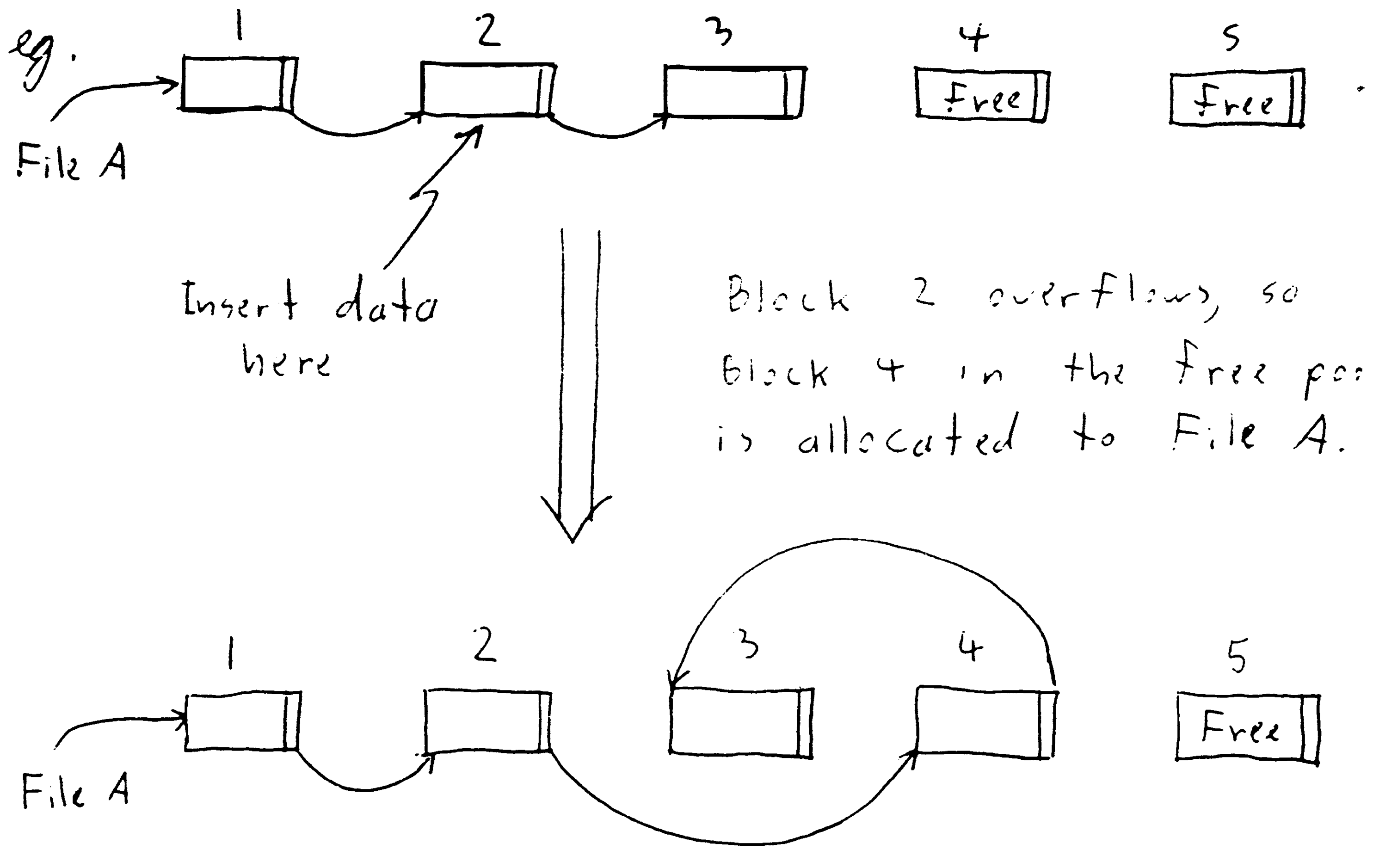
eg.



Can insert a record here without rewriting the other blocks

__Note:__ Occasionally, as the file expands, blocks will be taken from the free pool and allocated to the file.

eg.



Block 2 overflows, so Block 4 in the free pool is allocated to File A.

__Note:__ Block 3 is now the __4th__ block in the file.

Disadvantages

- To access a block in the middle or end of the File, all previous blocks must be accessed sequentially.

- ie, Can't **directly** access the blocks of a file.

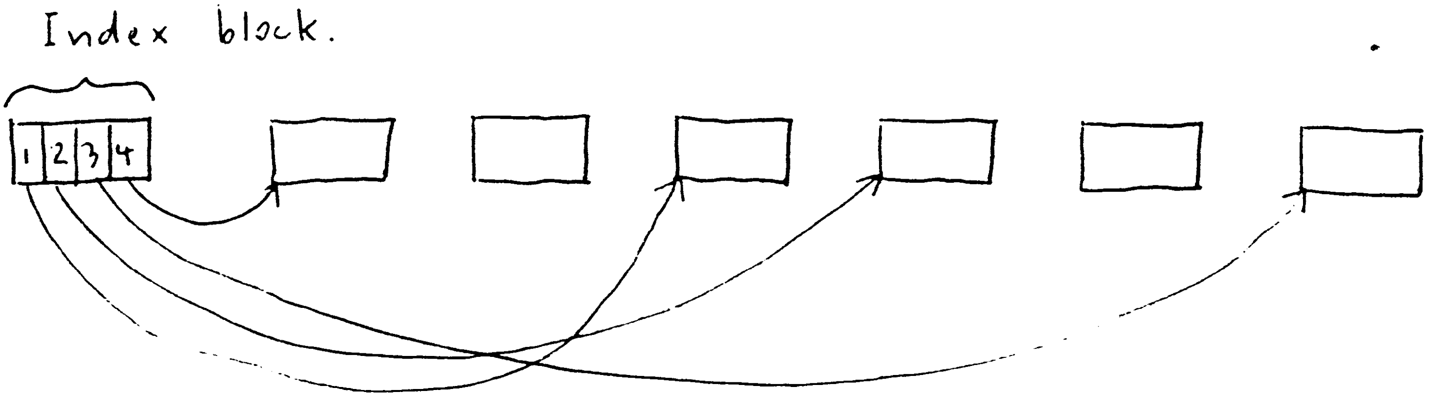
- File may be scattered over disk, so accessing all the blocks may involve wild seeks (movements of the disk head), which take a lot of time.

- Space must be reserved For a pointer in each block.

③ Indexed Allocation

For each file, keep an index of the blocks used by the file.

eg.

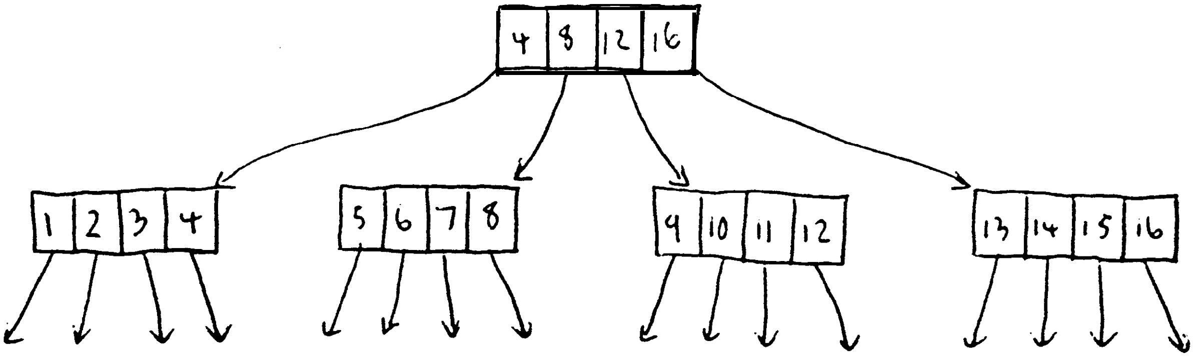


Question: What if the number of blocks in a file is greater than the number of pointers in a block?

Answer: Use multi-level indexing.

Multi-Level Indexing

eg.



Advantages:

- No external fragmentation.
- File can be modified without rewriting the entire file.
- Relatively easy to find any block in the file. ie, Allows direct access.

Disadvantages

- Space must be devoted to the index block
- Blocks may be scattered over disk.
- Finding the i^{th} block is slower than For segmented allocation.

Note:

- For most applications, two levels of indexing are enough.
- Unix uses a variation of indexed allocation, as we shall see.

File Directories

- A disk usually stores many files.

- For each file, the file system must keep track of certain information, such as
  - File Name
  - File Size
  - Start address of the file.

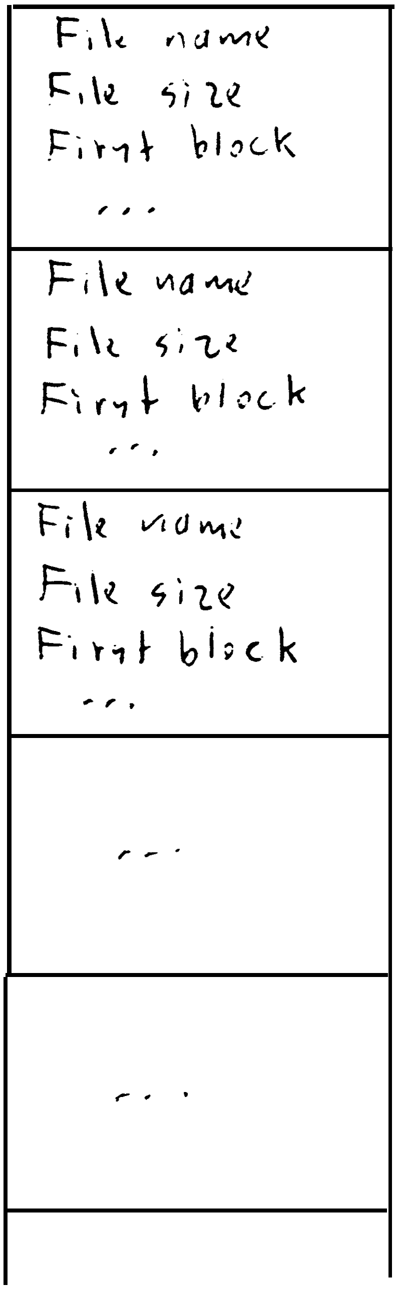
    eg. A printer to the first block in a linked list, or to an index block.
  - Ownership and access privileges.

    eg. Who may access the file.
  - Access information.

    eg. Who created the file, and when.
    Who last accessed the file, & when.

- This information is kept in a special system file called the <u>file directory</u>.

- The file directory has one record for each file on disk.



A File Directory.

File Directory Organization

- The File directory is kept on disk, for two reasons
  1. It is too big to fit in main memory
  2. It must survive system crashes, shutdowns, etc.

- When a user opens a file, the file system searches the file directory for the file's record.

- It then checks whether the user is allowed to access the file.

- If so, then it returns the start address of the file.

Performance Issue

- It would be very inefficient if the File system searched the file directory every time a user accessed a file.

- Fortunately, only a small number of files are open (being accessed) at a given time.

- The file directory entries for these files are kept in main memory, in the Active File Table (AFT).

- An entry is copied from the file directory to the AFT when the user opens the File.

eg. In Turing,

open: File number, File name

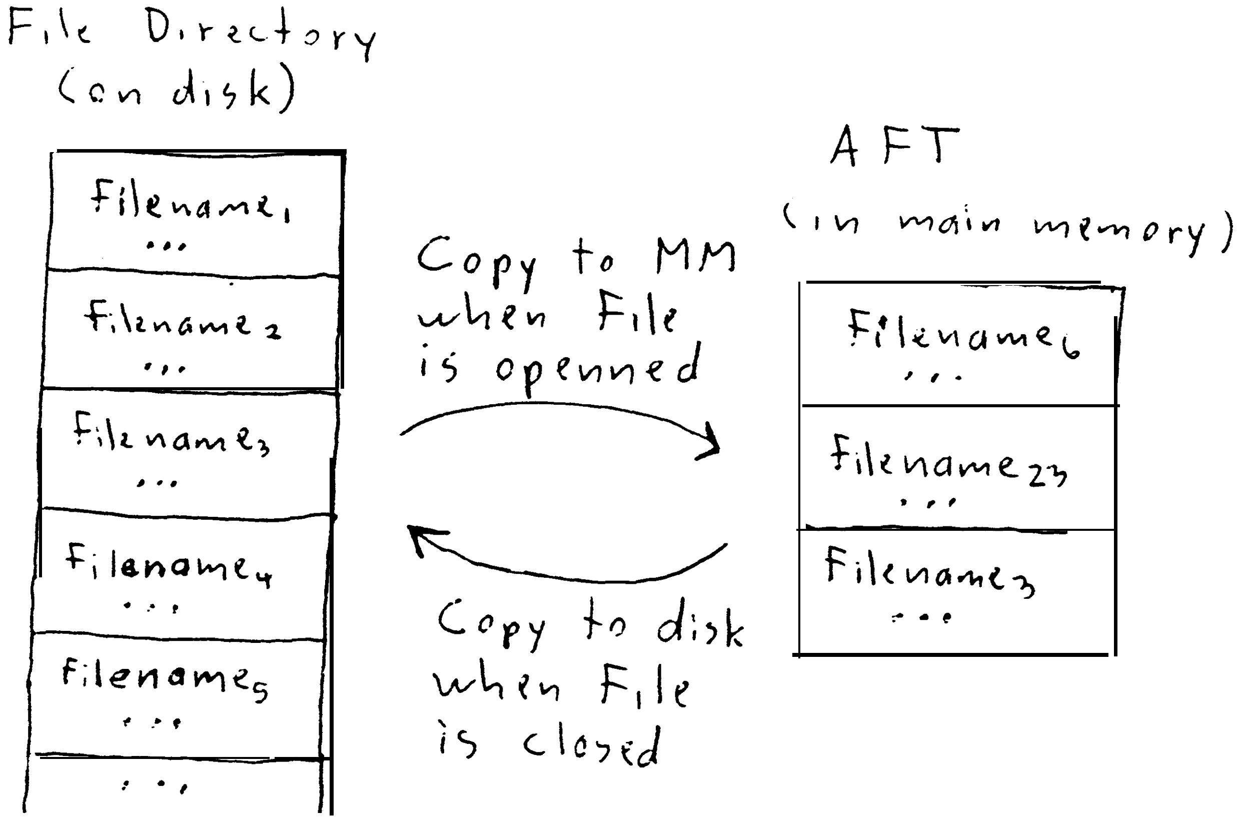
↑ user gives this

↑ File system returns this, a pointer into the AFT.

- If the user changes the file size or file access privileges, this is recorded in the AFT

- When the file is closed, the AFT entry is copied back to the file directory, on disk.

- Thus, the file directory is accessed only when a file is openned or closed.


File Directory
(on disk)
Filename$_1$
...
Filename$_2$
...
Filename$_3$
...
Filename$_4$
...
Filename$_5$
...
...
Copy to MM
when File
is openned
Copy to disk
when File
is closed
AFT
(in main memory)
Filename$_6$
...
Filename$_{23}$
...
Filename$_3$
...