Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Ecole supérieure en sciences et technologies de l'informatique et du numérique

General Information on Files

Presented by : Dr. Daoudi Meroua

Academic year: 2024/2025

Basic File Operations

Functional Processing:

a. Creation: Creating a file involves:

- Creating its structure, that is, defining its various fields as well as the length of its records.
- Entering the file's records and storing them on a magnetic (or optical) medium.

Basic File Operations

Example:

To create the Students file, its structure and the size of its records are defined as follows:

Field name	**Field length**
Student number	6 Character
Student fisrt name	10 Character
Student last name	10 Character
Date of birth	8 Character
Student address	20 Character

Then, we enter the information related to each student and save the file on the disk under the name: Students.

Basic File Operations

Deletion

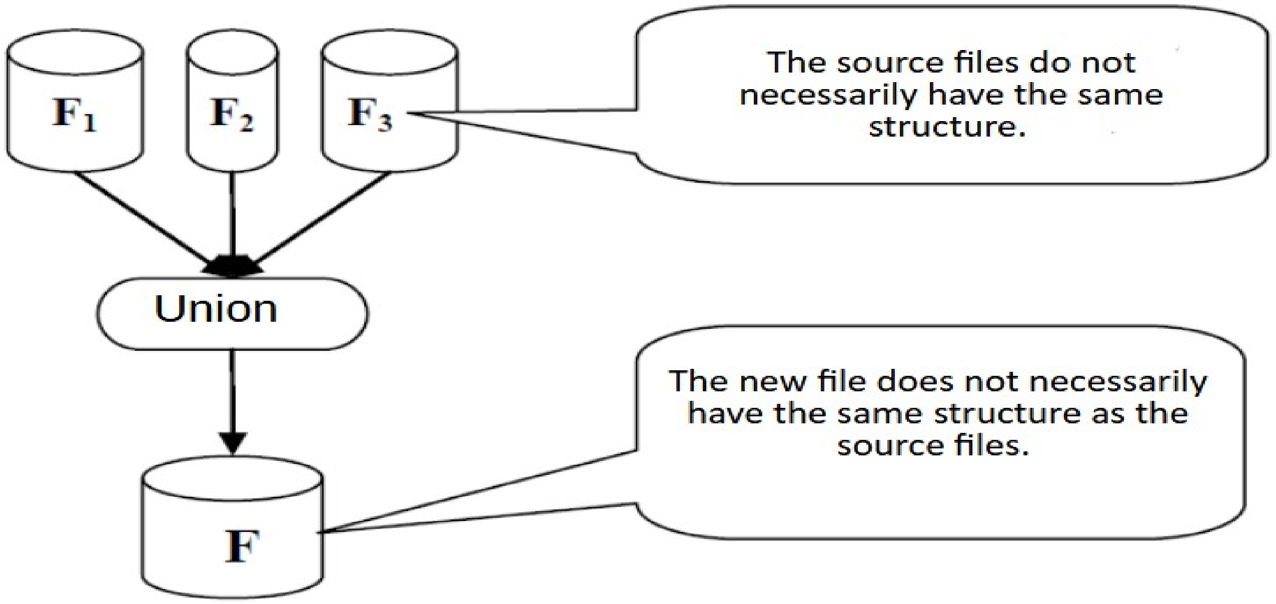
Deleting a file means canceling its storage, which involves erasing all the records that make it up, as well as its structure. There are two types of deletion: logical deletion and physical deletion.

Logical Deletion: Logical deletion consists of marking the file in such a way that it becomes transparent; in reality, it still exists on the storage medium.

Physical Deletion: Physical deletion permanently removes the file. The space previously occupied by the file will be reclaimed.

Basic File Operations

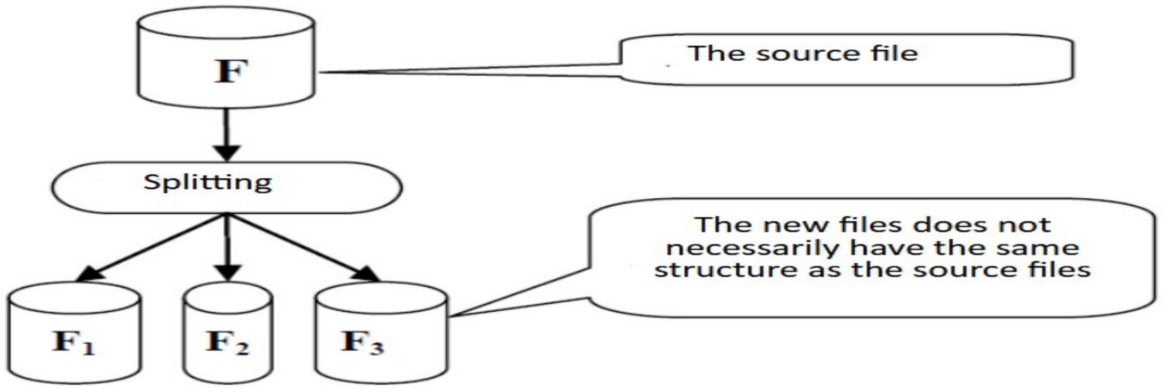
Union : Several source files give rise to a new file.


F1
F2
F3
The source files do not necessarily have the same structure.
Union
F
The new file does not necessarily have the same structure as the source files.

Basic File Operations

Splitting

This is the reverse operation of the union. A source file gives rise to several destination files.


F
The source file
Splitting
The new files does not necessarily have the same structure as the source files
F1
F2
F3

Basic File Operations

Sorting

The primary operation to perform on files is undoubtedly the search operation. To optimize the time for this search (to allow the user to access information as quickly as possible), it would be beneficial to store the information in a well-considered order. This organization operation is called sorting.

Sorting a file involves arranging its records in either ascending or descending order based on the value of one or more attributes known as sorting keys.

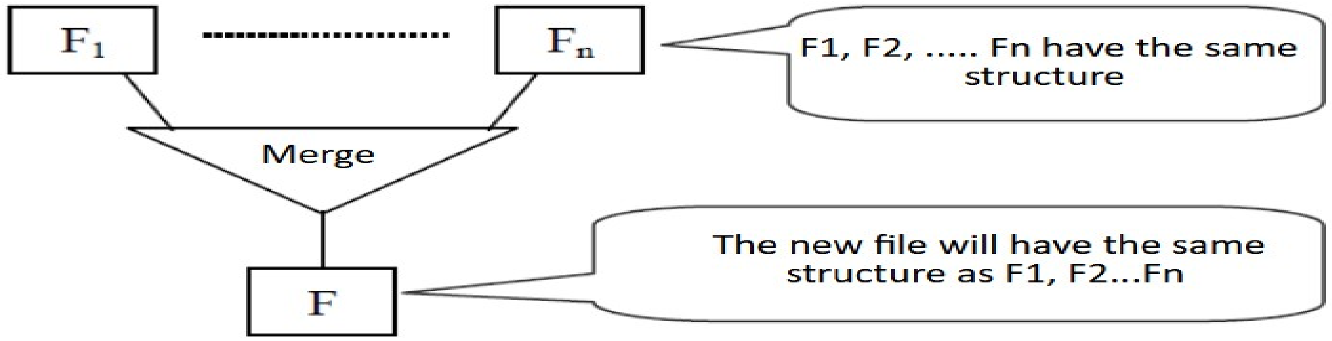
Basic File Operations

Merging

Merging involves combining the records of two or more files into a single file.

Condition: The files to be merged must have the same structure.

Consequence: The resulting file will have the same structure as the files that contributed to its creation.


F_1
F_n
F1, F2, ... Fn have the same structure
Merge
F
The new file will have the same structure as F1, F2...Fn

Basic File Operations

Updating encompasses the following three operations:

- Creating new records
- Deleting existing records
- Modifying the content of a record

Basic File Operations

Service Operations

Generally, three processes are distinguished:

- Copying
- Temporary storage
- Creating auxiliary files

Basic File Operations

Copying

Copying a file means duplicating its content onto a storage medium.

Temporary Storage

This process involves temporarily storing the results of a processing operation in intermediate files for later use. Their immediate use is delayed for one reason or another.

Creating Auxiliary Files

The creation of auxiliary files is related to understanding the characteristics of the equipment used and the constraints of the system to effectively utilize computing resources.

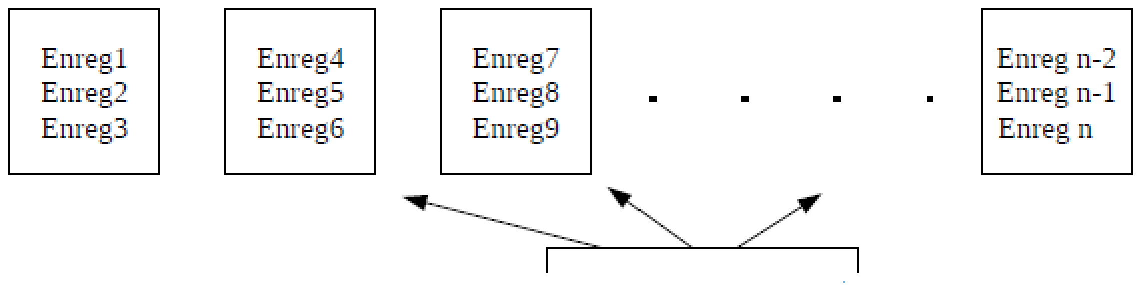
Text File and Binary File

A text file consists of a set of characters organized into lines, each capable of containing up to 255 characters, which end with a newline character. It can be opened by any text editor.

A binary file can contain all kinds of data in the form of a sequence of bytes, including text data. The bytes in a binary file are interpreted only by dedicated software; they are read and written as-is. A binary file can be structured (e.g., a database file) or unstructured.

File: Internal Level

- Regardless of the storage medium, a file = a set of physical blocks in a storage medium



A file = a set of physical blocks in MS

Physical File and Logical File

A logical file corresponds to the user's view of the storage of their data. More precisely, the logical file is:

- On one hand, a standard data type defined in programming languages, upon which a number of specific operations can be performed.
- On the other hand, a set of records or items.

Physical File and Logical File

A physical file corresponds to the entity allocated on permanent storage and physically contains the records defined in the logical file. The records that make up the logical file must be written into the sectors that comprise the blocks of the disk, thus forming the physical file corresponding to the logical file.

The physical file is therefore composed of a set of physical blocks that must be allocated to the file.

Physical Record vs. Logical Record

A logical record: The records of a logical file are referred to as logical records or items. The size of a record is measured in bytes or characters. There are two modes of dividing data into records:

- Fixed-length records
- Variable-length records

Exemple :

```
Struct etudiant {
                    char nom [20]
                    char prénom [20]
                     int num-inscription
                       }
```

Physical Record vs. Logical Record

A physical record: The records of a physical file are referred to as physical records.

The physical record represents the amount of information exchanged between main memory and the storage unit. A physical record or block is the smallest unit of data that can be read or written in a single operation. The physical record contains one or more records and possibly some control and organizational information.

The Blocking Factor

The number of logical records contained in a physical record corresponds to the blocking factor. Blocking a certain number of logical records into physical records primarily allows for a time gain during the execution of I/O operations. .

Memories

There are several types of storage media for manipulating and storing files, classified by:

- Access speed
- Cost of storing information
- Reliability

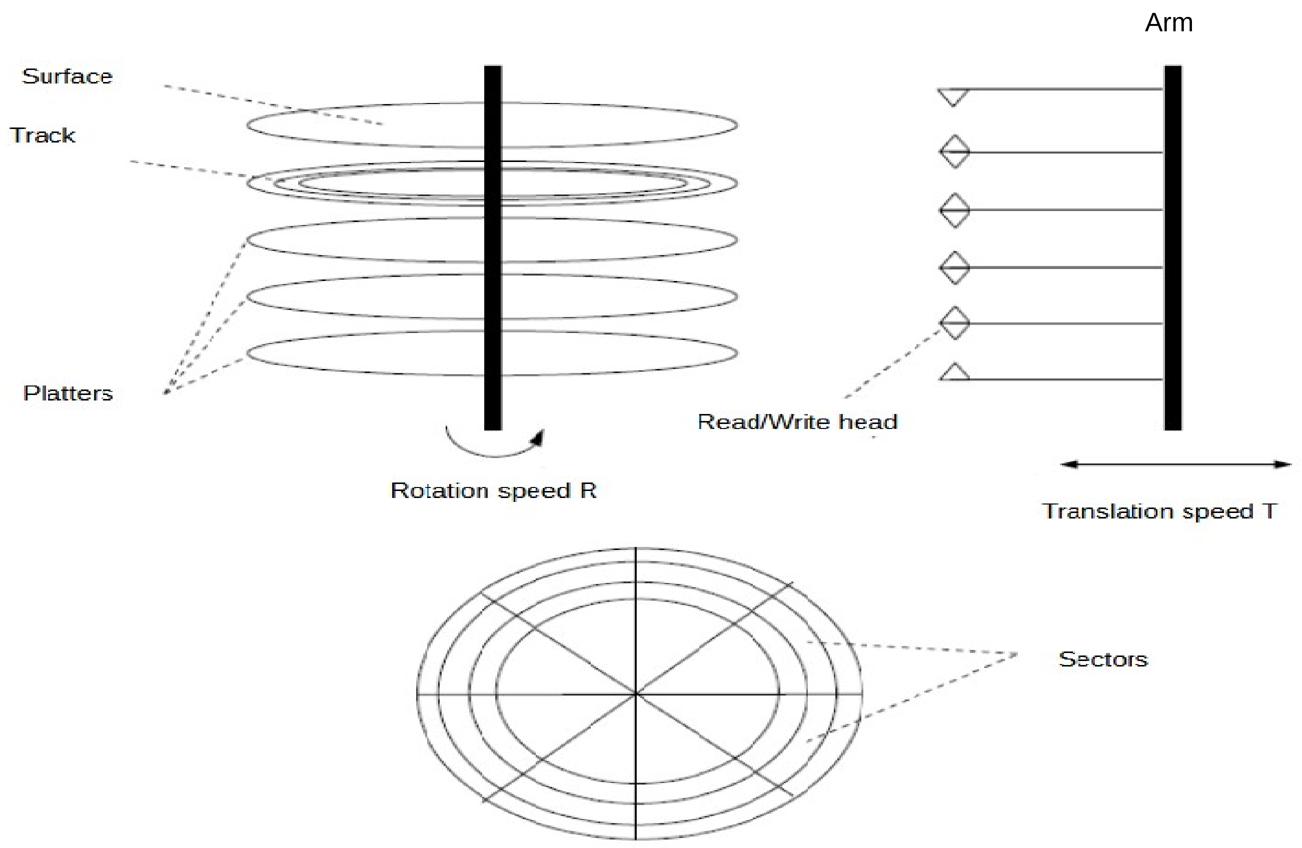
Examples: processor registers, cache memory, main memory, a buffer (between main memory and storage), secondary memory, and archival memory.

Memories

	Main Memory	**Secondary Memory**
Speed	Tens or hundreds of nanoseconds	A few milliseconds
Capacity	Limited	Large Capacity
Cost	High	low
Volatility	volatile	Non volatile

HDD

Les Disques Durs (HDD)



HDD

Disk Structure

- The disk is made up of a set of platters.
- Each platter consists of a set of tracks and sectors.
- The intersection of a track and a sector is called a block.
- Block Address = Sector Number + Track Number
- The address of a byte within the block represents its offset.
- Reference of a byte on the disk = Track Number + Sector Number + Offset

511

HDD

Reading a Block from the Disk

a) You must know the address of the block on the disk.

b) Position the read head on the correct track.

c) Wait for the correct block to come under the read head.

d) Transfer the block to main memory.

Time required for b) + c) + d): 8 to 9 milliseconds

Time for d): approximately 0.1 milliseconds

→ It is advantageous to read several contiguous blocks in succession.

HDD

Buffering

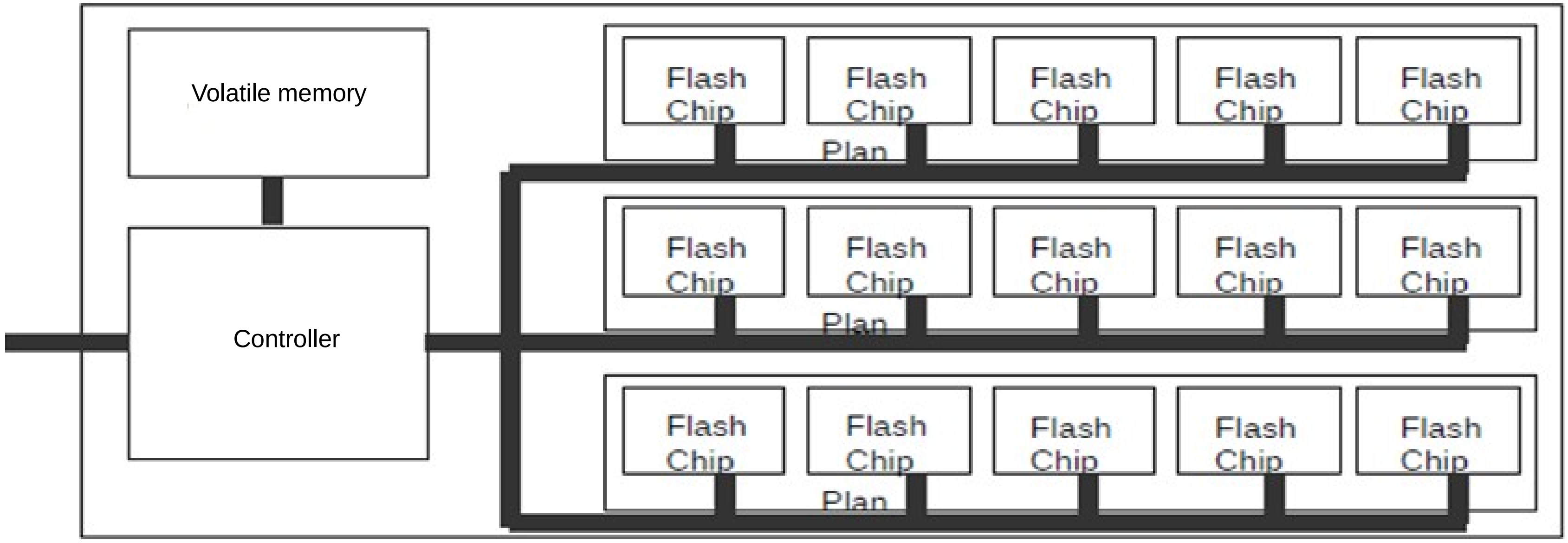
Buffering involves reserving a section of main memory that can hold multiple blocks at the same time (the buffer cache).

Prefetching

Prefetching, which consists of reading several consecutive blocks when a request is made for a specific block, is often used in combination with buffering to increase the chances of finding the requested blocks already present in main memory during any future accesses.

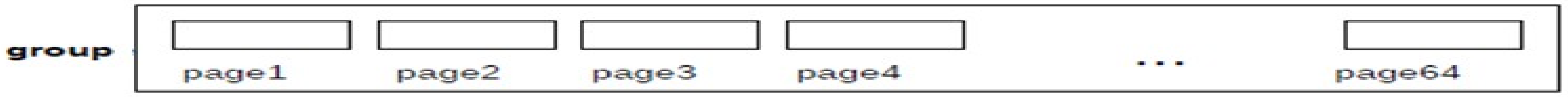
SSD

Solid State Drive SSD



SSD

The memory of the SSD is divided into groups (or blocks of pages, with a fixed capacity, e.g., 256KB), and each group is composed of a certain number of pages (e.g., page size = 4KB).



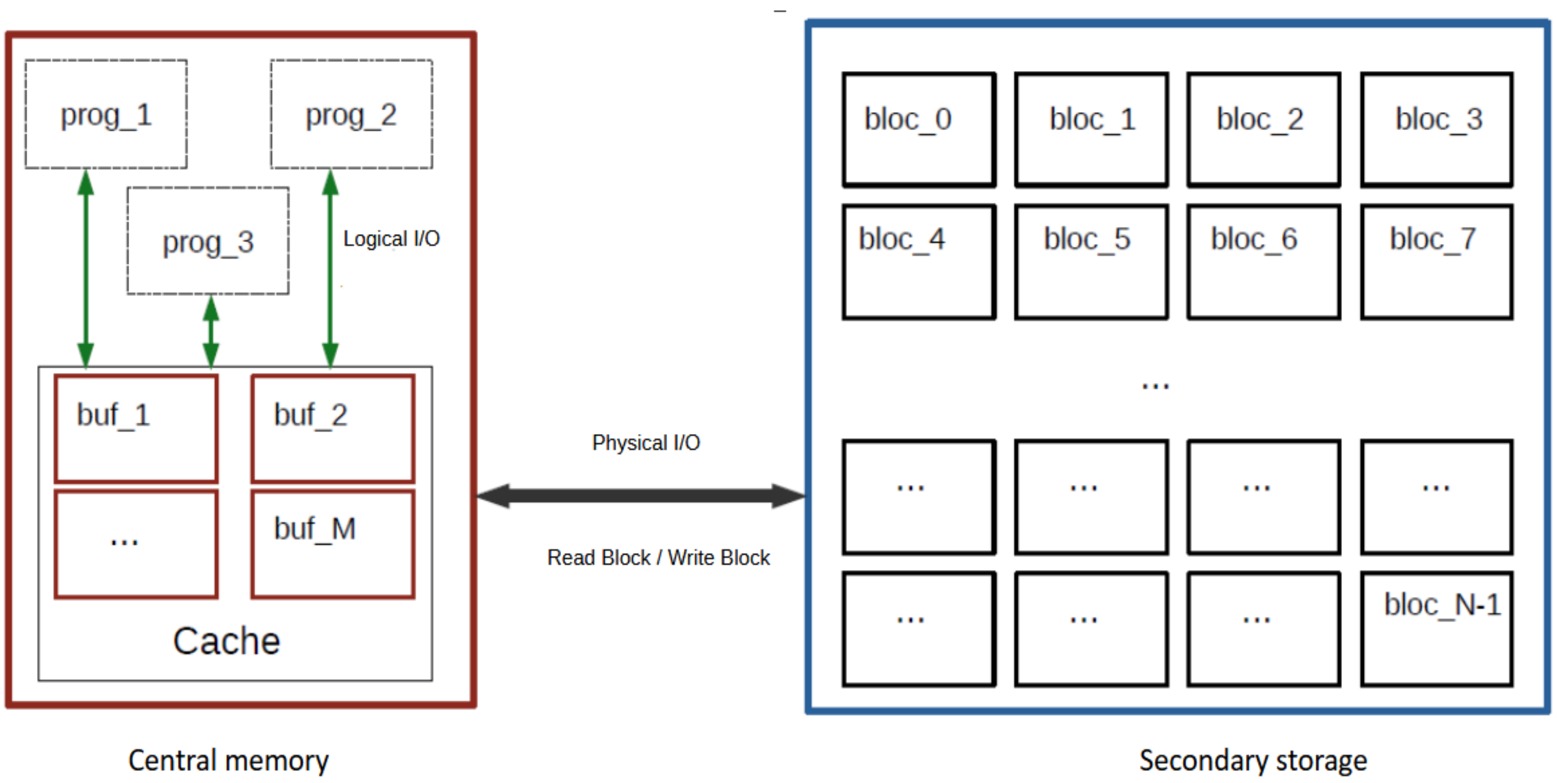
Read a page: very fast (20 microseconds)

Write a page, provided it is in an erased state: fast (100 to 200 microseconds)

Erase all pages of a block: slow (a few milliseconds)

MC/MS

Regardless of the type of disk (HDD or SSD), the interface remains the same:
The unit of transfer = a physical block (1+ sectors for HDD or 1 page for SSD)



Abstract Machine

When declaring a file, the type of blocks, buffers, and the characteristics to be used (header) will be defined.

Ex :

```
TypeBloc = struct
      tab : tableau[b] of TypeEnreg
      NB : entier
Fin

F : FICHIER of TypeBloc BUFFER buf1, buf2 ENTETE ( entier )
```

Abstract Machine

F : FICHIER of TypeBloc BUFFER buf1, buf2 ENTETE (entier)

In this declaration, it is stated that:

- F is a file composed of blocks with the structure TypeBloc.
- buf1 and buf2 are two variables (in main memory) of type TypeBloc (they will be used as buffers to access the blocks of F).
- The characteristics of F are summarized in a single integer (for example, it can be used to store the number of blocks in F).
- Buffer variables can also be declared independently of F (in separate declarations).

buf3 : TypeBloc ;

Abstract Machine

The operations of the model are:

OUVRIR(F , nomfichier , mode)

Open or Create a File

mode = 'A' *means open an existing file for reading/writing. The characteristics will be read into main memory upon opening.*

mode = 'N' *means create a new file for reading/writing. The characteristics will be allocated in main memory during creation***.**

FERMER(F) Close the file. The characteristics will be saved in secondary memory.

Abstract Machine

LireDir(F , i , buf) Read block number i of F into the variable buf.

EcrireDir(F , i , buf) Write buf into block number i of F.

ENTETE(F , i) Return the value of characteristic number i.

AFF_ENTETE(F , i , v) Assign v to characteristic number i.

ALLOC_BLOC(F) Return the number of a new block allocated to F.