INITIATION AU LANGAGE BASIC

TOME 2

C. CHICOIX Professeur d'université à l'INSA de Rennes

J. DEWITTE Ingénieur à l'INSA de Rennes

M. OLLIVIER Ingénieur à l'INSA de Rennes

1 NOTION DE FICHIER

Dans de nombreuses applications informatiques, il est nécessaire de conserver un grand nombre d'informations. Ainsi pour une entreprise qui veut gérer automatiquement ses commandes, il faudra pouvoir conserver des informations telles que :

- le nom et l'adresse de chaque client.
- les articles commandés par chaque client.
- la situation des comptes clients.

Cet ensemble d'informations pourra être stocké dans un (ou plusieurs) fichier.

Un fichier est défini comme une collection d'informations enregistrées sur un support de mémoire autre que la mémoire centrale d'un calculateur et défini par un nom.

Le support peut être :

- de type papier (ruban ou cartes perforées - listing...)
- de type magnétique (disque ou bande).

Un fichier peut contenir :

- des données nécessaires à l'exécution d'un programme.
- le programme lui-même, quand il est stocké sur mémoire secondaire.

Un fichier de données contiendra souvent (surtout en gestion) des informations de même nature regroupées en enregistrements (ou articles) de composition connue.

A la création d'un tel fichier, le programmeur devra définir la composition de l'article et éventuellement sa longueur.

Exemple : un fichier d'adresses peut avoir la composition suivante :

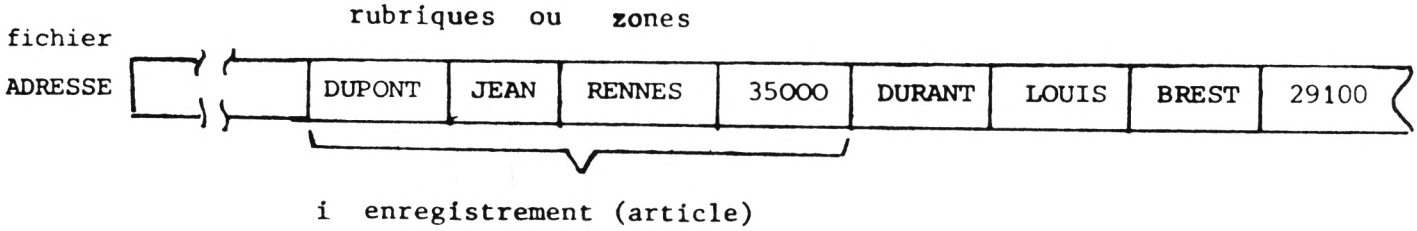
NOM

PRENOM

ADRESSE

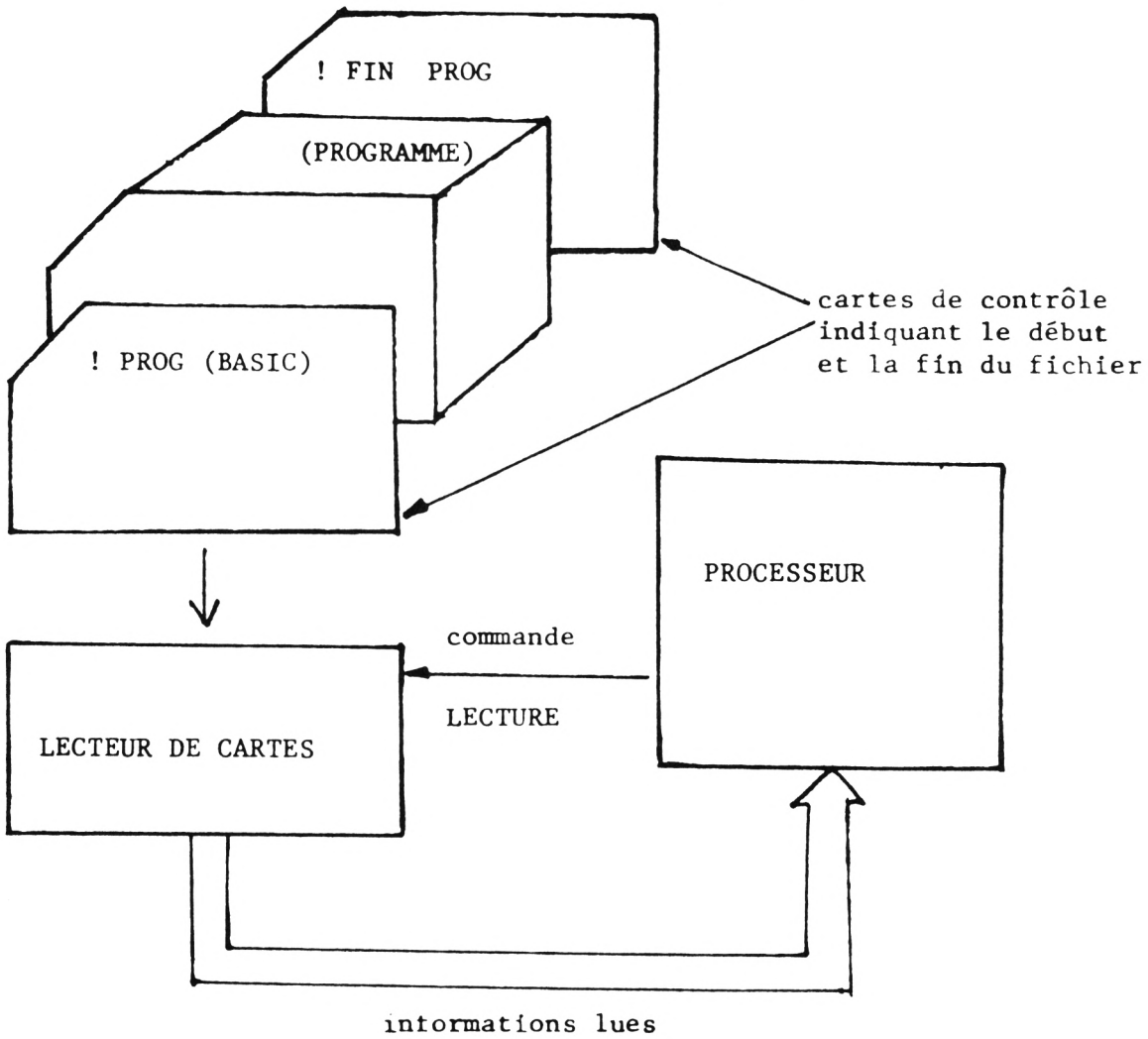
CODE POSTAL

Les informations pourront être rangés comme suit :



II LES SUPPORTS DE FICHIERS

Ce que l'on demande à un support de fichier est de pouvoir stocker des informations susceptibles d'être relues par un processeur. Les plus anciens utilisés consistent en un support papier, qu'il s'agisse de cartes ou de ruban perforé. Ainsi un paquet de cartes perforées comportant les lignes d'un programme constitue un fichier qui sera lu (ou exploré) lorsque l'on demandera l'exécution du programme.



Bon nombre de problèmes vont se poser avec un tel support (ces remarques sont pratiquement valables qu'il s'agisse de cartes ou de ruban):

d'épaisseur correspond au maximum à environ 40 000 caractères)

- vitesse à laquelle les informations peuvent être stockées ou relues : vitesse de transfert. Les ordres de grandeur sont de 10 000 caractères par minute en entrée, 1000 caractères/min en sortie.

-Possibilités de manipulation du fichier. Si l'on veut relire le programme (réexploration du fichier), on devra replacer manuellement celui-ci dans le lecteur de cartes. Si l'on veut modifier le programme (réactualiser le fichier), on devra changer manuellement les cartes concernées, ou bien faire reperforer automatiquement l'ensemble du programme (à condition de disposer d'un perforateur de cartes couplé au processeur). On peut signaler au passage que l'information sur support papier n'est pas modifiable (impossibilité de réécriture sur un support déjà utilisé).

On voit apparaître ici les différentes qualités que l'on demandera à un support :

- capacité de stockage d'une unité (par exemple le bac à cartes)
- volume du stockage résultant pour un fichier donné
- vitesse de transfert des informations
- possibilité de réactualisation et réexploration du fichier.

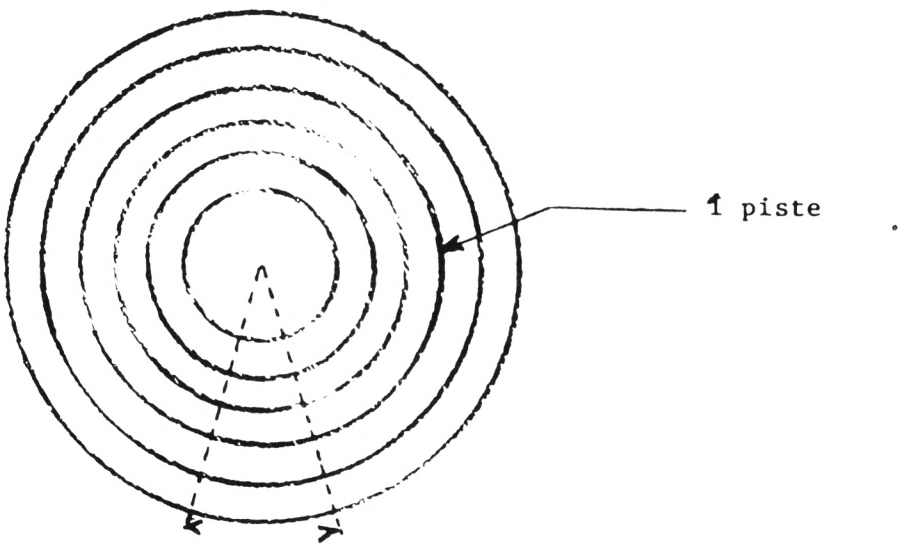
Les supports magnétiques tendent à éliminer complètement le support papier, car répondant mieux à ces différents critères. L'information, au lieu d'être stockée sous forme de trous (dont la présence ou l'absence signifie "1" ou "0" à la relecture) figure sous la forme d'un enregistrement magnétique (par exemple une modulation de fréquence : à une fréquence f_1 correspond un "0", à une fréquence f_2 un "1"). On peut distinguer deux grands types de supports magnétiques, bandes et disques, dont les intérêts sont différents.

Tous deux sont réutilisables, le même support physique pouvant ainsi servir à l'enregistrement et à la relecture, à l'aide d'un système mécanique unique.

Les bandes magnétiques constituent encore des supports de type séquentiel. Cela signifie que l'accès à une information contenue sur la bande nécessite de "dérouler" l'ensemble de la bande qui la précède. Le temps de transfert proprement dit sera donc précédé d'un temps d'accès

dépendant de la longueur de bande à passer et de la vitesse de défilement. L'intérêt des bandes magnétiques est leur très grande capacité de stockage. Les systèmes professionnels permettent souvent des densités de 1600 BPI (Bit per inch), avec des vitesses de transfert de l'ordre de 100 000 caractères par seconde. Ces systèmes sont donc très adaptés à la sauvegarde de gros fichiers. Cependant le prix de ces périphériques est tel qu'on ne les trouve pratiquement pas sur microordinateur. On y couple plutôt des systèmes à cassettes. Ces dernières peuvent être de type digital (c'est-à-dire avec la même organisation que les bandes magnétiques, mais une densité moindre et une vitesse de transfert de l'ordre de 1000 car/s) ou bien de type audio. Ce sont plutôt ces dernières que l'on trouve sur microordinateur, pour des questions de prix (les cassettes digitales tendent à être remplacées par les disques souples,que nous verrons plus loin, plus simples d'emploi pour un coût comparable). Ce coût très bas (300 F 1981 un lecteur-enregistreur, 10 F 1981 une cassette) est malheureusement accompagné d'une vitesse de transfert faible (30 car/s) et d'une capacité plus réduite (54 K octets/face pour une C 60, le temps de chargement d'un tel fichier étant d'une demi-heure...). Ces caractéristiques en font plutôt un système de stockage réservé aux utilisations amateurs.

Les disques magnétiques constituent, eux, des supports dits à accès aléatoire. Ce dernier terme est à prendre dans le même sens que pour les mémoires à accès aléatoire : chaque information possède une adresse et, on pourra accéder à chaque information, en lecture ou en écriture, simplement en spécifiant son adresse, le système de lecture-enregistrement se positionnant directement sur le bloc considéré. Les informations sont enregistrées sur le disque sur des pistes concentriques, chaque piste étant divisée en secteurs.



C'est l'ensemble "N° de piste, N° de secteur" qui constitue l'adresse de l'information. Cette adresse pointe sur un bloc dont la taille est de plusieurs mots mémoire du processeur (par exemple 128 octets), ces mots étant enregistrés sous forme séquentielle à l'intérieur du bloc. La capacité d'un disque sera alors :

NB Secteurs * NB Pistes * NB mots par Secteur

On distingue deux types de disques : disques durs et disques souples. Les disques durs sont constitués d'un support rigide recouvert d'oxyde magnétique. Ils autorisent des positionnements mécaniques très précis, et permettent des vitesses de rotation élevées.

Ainsi, le temps d'accès aux informations peut être faible, le secteur demandé repassant souvent sous les têtes de lecture-écriture. Ce temps (de l'ordre de 10 MS) peut cependant être augmenté si les têtes n'étaient pas positionnées sur la bonne piste (ce déplacement peut prendre jusqu'à 100 MS). La vitesse de transfert est alors d'environ 1 M car/s. Le disque peut être amovible ou bien fixe. Ce sont plutôt ces derniers, dits" WINCHESTER", que l'on trouve sur les microordinateurs du fait de leur prix (50 000 f 1981 pour 20 M octets).

Cependant le support le plus couramment disponible actuellement est constitué d'unité de disques souples (floppy disk). Du fait de leur mécanique simplifiée (les têtes reposent sur le disque, qui en contrepartie tourne moins vite), leur prix est très intéressant (4000 F 1981 la platine). Il existe différents modèles, pouvant aller de 80 K octets à 500 k octets stockés par disquette, le principe de fonctionnement étant identique à celui des disques durs. Par contre, du fait de la vitesse de rotation plus lente, les temps d'accès sont allongés (environ 50 MS plus le temps de positionnement de la tête, qui peut atteindre 1 s). La vitesse de transfert est également plus faible (de l'ordre de 40 000 car/s).

D'autres types de support apparaissent, dont il est difficile de prévoir l'évolution et la part qu'ils prendront. Signalons :

- les mémoires à bulles. Ils s'agit de mémoires de type magnétique , directement incorporables dans l'espace mémoire du système. Il existe des boitiers, de la taille d'une petite boite d'allumettes, d'une capacité d'un million de bits.

- les supports "optiques". Il s'agit du vidéo-disque, qui pourra réaliser un stockage de très grosse capacité. Par contre, il

s'agira d'un support non réutilisable (une fois le disque préssé, l'information ne peut être modifiée).

III ACCES AUX FICHIERS

La méthode d'accès à un fichier est la façon dont celui-ci peut être traité par un programme.

Il existe en fait seulement deux types d'organisation de fichiers :

- L'organisation séquentielle : pour lire un enregistrement du fichier, il faudra lire tout ce qui précède.

- L'organisation directe : en positionnant directement la tête de lecture/écriture sur l'endroit où se trouve l'enregistrement recherché, on accède directement à celui-ci.

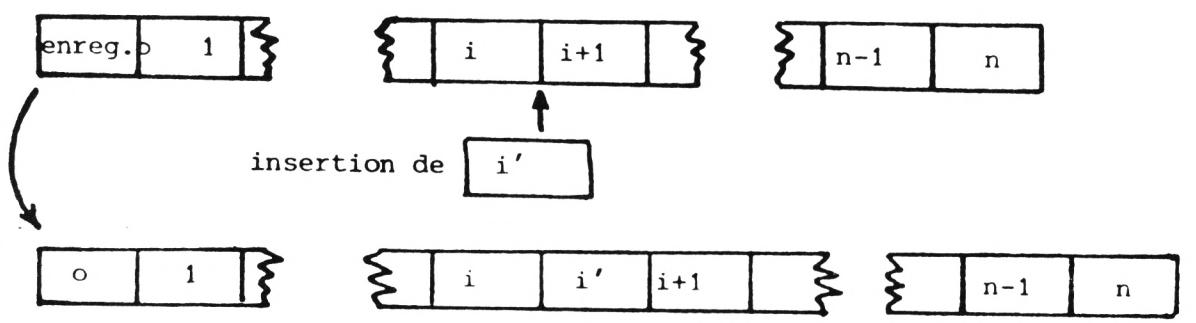
a) Les fichiers à accès séquentiel

Ce type de fichier est le plus simple à utiliser. C'est aussi celui qui offre le moins de possibilités de travail.

Si l'on considère un fichier séquentiel des clients d'une entreprise, la recherche d'un client particulier se fera en lisant séquentiellement tous les enregistrements qui précèdent. Le temps d'accès à un enregistrement peut donc être très long.

EXEMPLE : si l'on veut accéder au 1000 ème enregistrement d'un fichier séquentiel et que le temps d'accès à un enregistrement est $1/10^{e}$ de seconde, il faudra attendre 100 secondes !

De même lorsqu'on veut ajouter des données sur un fichier séquentiel, il faudra se positionner à la fin du fichier. Ceci veut dire que pour insérer un enregistrement dans ce type de fichier à une place bien déterminée, il faudra le lire entièrement et le recopier après avoir fait l'insertion.



Certains systèmes de gestion considèrent même que tous ajout (y compris en fin de fichier) est une "insertion".

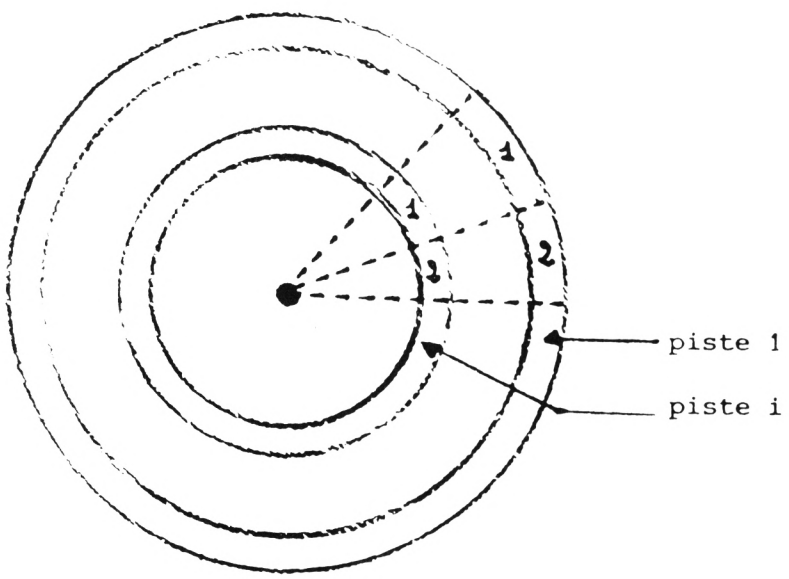
En conclusion :

- Ce type de fichier n'est pas adapté au travail en temps réel. Il ne permet pas la mise à jour immédiate d'enregistrements
- Il se prête à des traitements séquentiels tels que sortie d'étiquettes ou édition de bulletins de paye.
- Ces fichiers sont les seuls utilisables sur des supports séquentiels tels que bande magnétique ou cassette.

b) Les fichiers en accès direct

Pour mettre en oeuvre ce type de fichier, le dispositif de stockage doit être un disque ou une disquette magnétique.

Intéressons-nous tout d'abord à la structure d'une disquette. Celle-ci est divisée en pistes concentriques, chaque piste contenant un ensemble de secteurs. Chaque secteur peut contenir un nombre d'octets bien déterminé (souvent 128, 256 ou 512 octets)



Le dispositif de commande permet de positionner la tête de lecture sur une piste déterminée.

Il est donc possible d'accéder à une information (contenue dans un ou plusieurs secteurs) sans lire tous les enregistrements des pistes qui précèdent. On parle alors d'accès direct à l'information.

Dans la pratique, l'utilisateur connaît pour un enregistrement donné le numéro d'ordre ou la clé de ceui-ci relativement au début du fichier qui le contient.

Il accèdera donc au 1000^{e} enregistrement d'un fichier client par un ordre de la forme :

LIRE DANS FICHIER 6 CLIENT ENREG 1000

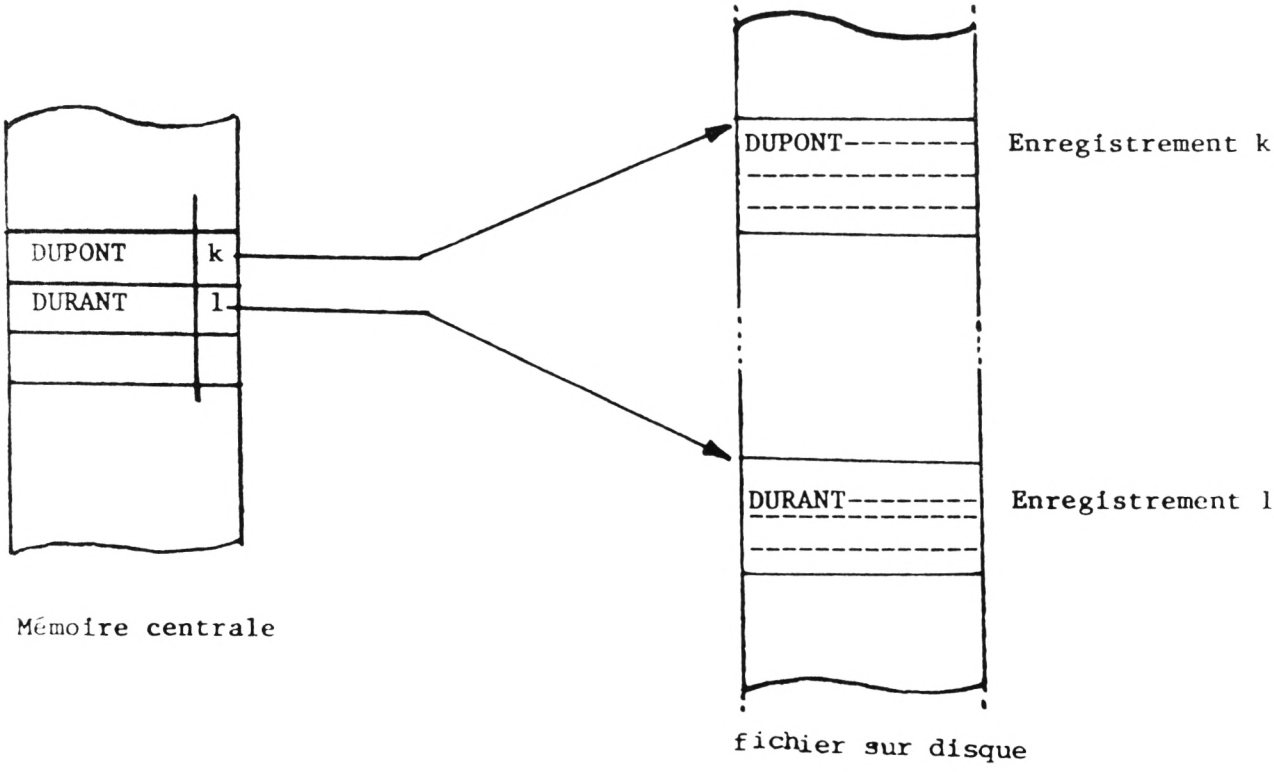
Le système de la machine, connaissant l'emplacement du fichier sur le disque calculera automatiquement l'adresse du secteur contenant l'information recherchée.

Ce type d'accès est donc tout à fait adapté si l'on veut un accès rapide aux données d'un fichier.

c) Les fichiers en accès indexé

Cette méthode est en fait un accès direct d'un type particulier.

Elle utilise une table en mémoire qui à chaque clé associe la place de l'enregistrement correspondant sur le disque. Cette table s'appelle table d'index ou table de correspondance.



Pour accéder à un enregistrement sur le disque il faut donc chercher tout d'abord, en fonction de la clé d'accès, sa place dans le fichier à travers la table d'index.

Cette méthode permet un accès très rapide aux informations du fichier. Par contre si ce fichier contient beaucoup d'enregistrements, la table de correspondance peut être très grande et par conséquent occuper une grande partie de la mémoire centrale du calculateur.

Pour pallier à ce défaut, dans certains cas, la table est elle - même un fichier sur disque, qui peut être exploré plus rapidement que le fichier principal grâce à sa plus petite taille. Les clés seront dans ce cas rangées par ordre croissant dans la table de correspondance.

D'autres fichiers d'index peuvent être créés. Ils permettent alors d'accéder au fichier principal à travers d'autres clés.

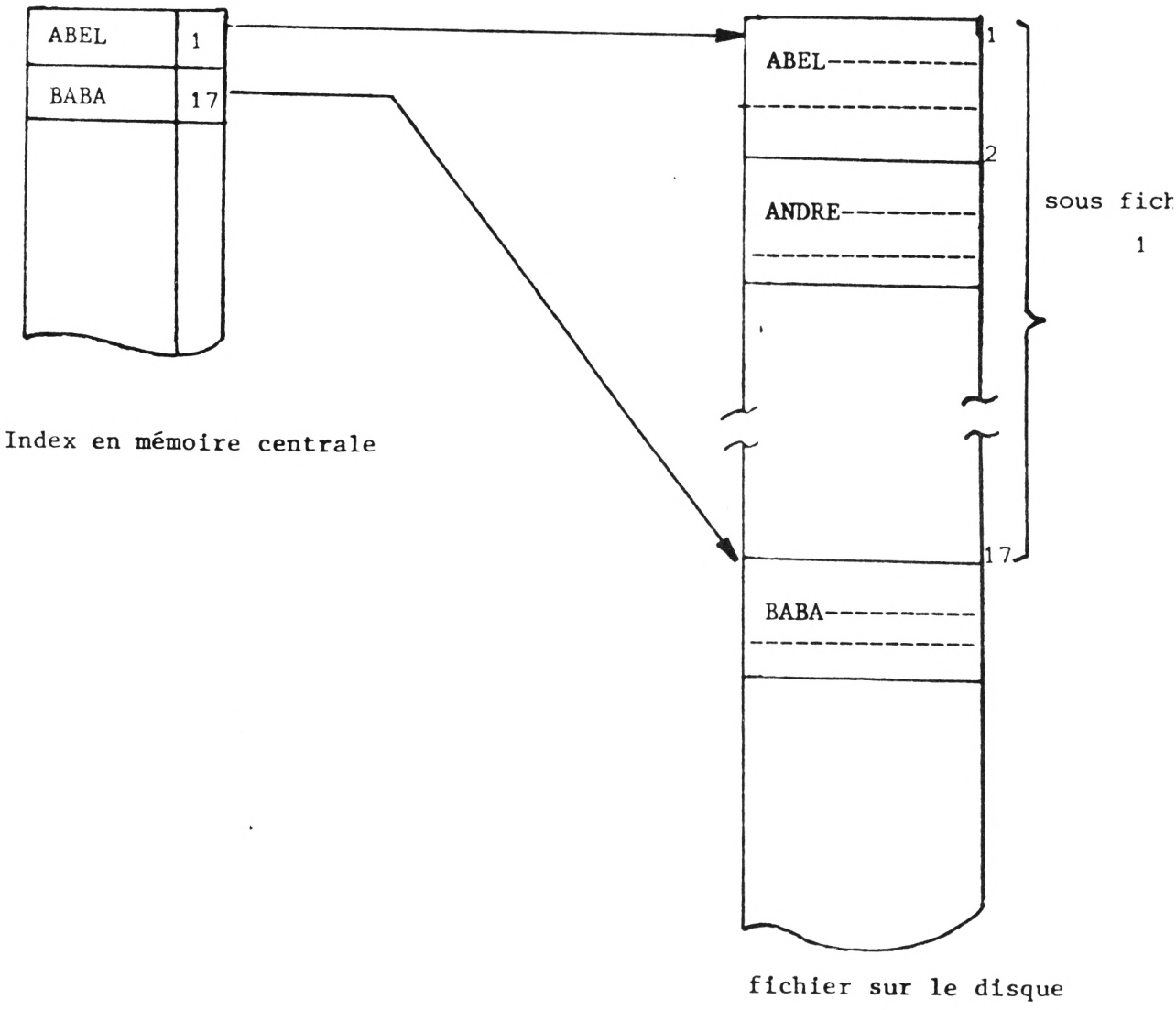
Dans le cas de fichiers à accès indexé, l'ajout d'enregistrements ne pose pas de problème . En effet ceux-ci pourront être placés en fin de fichier. Par contre il sera nécessaire de mettre à jour la table de correspondance en insérant la clé du nouvel enregistrement à sa place. Si le système a été mal conçu, une réorganisation totale de cette table sera nécessaire. Pour éviter ce travail, des places vides seront prévus dans la table. Ces trous permettront l'insertion des nouvelles clés sans problème. Une réorganisation interviendra quand tous les trous auront été utilisés.

d) Les fichiers en accès séquentiel indexé

Cette méthode d'accès ressemble beaucoup à la précédente. La nuance tient dans le fait que l'index ne correspond pas à une valeur de clé isolée mais à un groupe de clés. L'index en précise alors le premier et le dernier élément. Il pointe donc sur un sous-ensemble du fichier.

Les enregistrements sont rangés par ordre croissant des clés dans le fichier. La table d'index se trouve en mémoire centrale.

Exemple :



L'accès à un enregistrement se fera donc de la manière suivante :

- recherche dans la table de l'index.
- recherche séquentielle de l'enregistrement à l'intérieur du sous-fichier pointé.

L'insertion d'une donnée dans le fichier ne posera pas de problème dans la mesure ou le programmeur aura laissé des "trous" dans chaque sous-fichier.

Cette méthode d'accès est très puissante mais complexe à mettre en oeuvre. Il arrive que sa gestion soit effectuée automatiquement par le système de gestion de fichiers de la machine.

IV LES FICHIERS EN BASIC

Nous avons vu les différentes organisations des fichiers. Le problème qui se pose est de savoir quelles possibilités offre BASIC. A cette question il est impossible de faire une réponse unique : il y a autant de systèmes de gestion de fichier que de réalisations de BASIC . Il existe cependant une constante : tous les BASIC autorisent la gestion de fichiers de type séquentiel. La raison en est simple : il est nécessaire de pouvoir stocker et recharger des programmes, sans être obligé de les rentrer au clavier à chaque nouvelle séance de travail. Dans les systèmes de bas de gamme, ces possibilités sont liées à l'utilisation d'un système utilisant des cassettes audio comme support de fichiers.

Cependant les systèmes qui nous intéressent sont ceux capables de gérer un disque, c'est-à-dire munis d'un "Disk Operating System" (D.O.S.). Ces systèmes de gestion de fichiers existent obligatoirement sur tous les microordinateurs munis de disques souples ou durs. Il faut signaler dès maintenant qu'il ne s'agit là que de systèmes de gestion de fichiers, et non de base de données, et que certains BASIC se limitent, même sur disque, à la gestion des fichiers de type séquentiel.

On peut distinguer deux niveaux dans un D.O.S. :

- commandes de gestion du disque. Ces commandes permettent principalement de manipuler les fichiers contenant les programmes.

- commandes d'entrée-sortie sur fichier disque. Ces commandes permettent de sauvegarder et d'accéder à des informations de nature quelconque stockées sur disque.

L'utilisation d'un DOS va être illustrée à l'aide de celui du BASIC microsoft. Il est à noter que, même pour ce BASIC, il existe des nuances dans les possibilités et la syntaxe selon la machine où il est implanté.

Un disque peut contenir un nombre quelconque de fichiers (en fait ce nombre est limité comme nous le verrons plus loin). Aussi

ces fichiers doivent-ils pouvoir être identifiés. Chacun d'eux portera un nom qui permettra de l'appeler. Dans le cas qui nous intéresse, le nom sera constitué d'une chaîne pouvant comporter jusqu'à 8 caractères. A chaque fichier correspond un emplacement sur le disque, emplacement géré de manière automatique par le disk operating system. L'ensemble de ces noms de fichiers et de leurs emplacements physiques est regroupé en une table des matières appelée "Directory". Cette table des matières comporte un nombre maximal de lignes, ce qui limite le nombre de fichiers que peut gérer le système. La plupart du temps, au nom du fichier sont attachés un certain nombre d'attributs, qui en précisent la nature. Ici, les attributs sont gérés par le DOS, sans que l'utilisateur s'en préoccupe.

Nous avons vu qu'un disque s'adresse par secteur, ceux-ci correspondant à des enregistrements de longueur fixe.

Or les enregistrements manipulés (par exemple les lignes d'un programme) n'auront pas obligatoirement la même longueur qu'un secteur du disque. Le DOS utilise donc une mémoire-tampon (un Buffet) par laquelle transiteront les informations à échanger entre un programme et le disque. Si l'on désire manipuler des informations sur plusieurs fichiers à la fois, il sera nécessaire de disposer d'autant de "Buffe " que de fichiers en cours de traitement. Afin d'économiser la place mémoire (chaque Buffet prend 135 octets dans le cas qui nous intéresse), ces buffet sont assimilés à des canaux de transmission. Un fichier à traiter sera alors associé à un numéro de canal donné, tous les transferts, entrée ou sortie, étant ensuite effectués en utilisant ce numéro de canal. Lorsque le traitement sera terminé, il sera possible de restituer le canal qui pourra alors être associé à un autre fichier. Le nombre de canaux que peut gérer le DOS Microsoft est indiqué par la réponse à la question "How Many Files ?" posée lors du lancement de l'interpréteur BASIC. Ce nombre ne peut être supérieur à 8 et est pris égal à 4 par défaut.

Un DOS est généralement capable de gérer plusieurs disques. Aussi il sera nécessaire d'indiquer sur quel disque se trouve le fichier sur lequel on veut travailler (par contre cela autorise d'avoir des fichiers de nom identiques sur plusieurs disques). D'autre part à chaque disque va correspondre une table des matières. Afin d'accélérer les recherches, la "directory" est parfois lue en début de session et stockée en mémoire centrale, et renvoyée sur le disque lorsque l'on cesse de travailler. Ce renvoi est nécessaire si des modifi-

cations ont été apportées aux fichiers contenus sur le disque, afin que lors de la session suivante, la "directory" soit en concordance avec le contenu réel du disque. Afin de savoir combien de directory il peut avoir à gérer, le BASIC microsoft demande, à son initialisation, le nombre de disques que l'on désire utiliser (ce nombre est pris égal à 2 par défaut, soit les disques de numéro 0 et 1).

Dans les paragraphes suivants, on respectera les notations :

⟨nom de fichier⟩	chaîne de caractère représentant le nom d'un fichier.
ND	numéro de disque (compris entre 0 et la réponse à "Highest disk number ?"). Par défaut, ce numéro sera pris égal à 0 dans les instructions d'entrée-sortie.
NC	numéro de canal (compris entre 1 et la réponse à "How many files ?").

Une dernière remarque : l'intérêt de ce Basic, est d'être entièrement résident en mémoire centrale, y compris, en ce qui concerne son DOS [illegible]. Contrairement à beaucoup d'autres, il ne fait pas appel à la segmentation (il s'agit d'une technique qui consiste à ne charger en mémoire centrale que la partie du programme qui va être exécutée, les segments n'étant appelés qu'au fur et à mesure des besoins). Ainsi la disquette ayant servi au chargement du BASIC peut être retirée de l'unité de disque et remplacée par une autre disquette ne comportant que des fichiers. Si cela améliore en outre les temps d'exécution (il n'y a pas de segment de programme à aller chercher sur le disque), par contre l'espace mémoire disponible à l'utilisateur se trouve réduit.

IV I Commandes de gestion des disques

Ces commandes permettent de manipuler les fichiers contenant les programmes et dans certains cas, les fichiers utilisateurs.

Afin de pouvoir travailler, certaines commandes sont absolument nécessaires :

- initialisation d'un disque. Cette commande permet entre autre d'initialiser la directory. Elle a pour effet de détruire toutes les

informations que pouvait contenir la disquette, et de rendre cette dernière utilisable par le DOS.

syntaxe : DSKINI ND

- charger en mémoire la directory d'un disque. Si, comme ici, le DOS travaille sur la directory uniquement en mémoire centrale, celle-ci doit être lue avant tout transfert : il faut "monter" la disquette.

syntaxe : MOUNT [ND [, ND]*]

Si ND n'est pas spécifié, les disquettes de toutes les unités spécifiées à l'initialisation sont montées.

- recopier la directory sur le disque. En cas de travail sur les fichiers, la table d'occupation de la disquette a pu être modifiée. Il est donc nécessaire de remettre à jour la table des matières qui se trouve sur le disque. Une sauvegarde d'un fichier ne sert à rien tant que la table des matières du disque n'est pas réactualisée, et des erreurs catastrophiques peuvent résulter de l'oubli de cette réactualisation.

syntaxe UNLOAD [ND [, ND]*]

Si ND n'est pas spécifié, toutes les disquettes montées sont démontées. Les fichiers figurant sur les disquettes démontées ne pourront pas être accédées de nouveau, avant que les disquettes soient bien montés de nouveau.

- sauvegarde d'un programme. Lorsqu'un programme a été entré au clavier, il est nécessaire, si l'on désire s'en servir plus tard, de le stocker.

syntaxe SAVE <nom de fichier> [ND [, "A"]]

Le programme sera sauvegardé sous le nom de fichier indiqué. S'il existe déjà un fichier de ce nom, il sera effacé et remplacé par le nouveau. Le programme est stocké sur le disque indiqué, 0 par défaut. Le stockage se fait normalement sous forme compactée. Cependant, il peut être nécessaire, dans certains cas, de le stocker sous forme de texte (suite de caractères ASCII). Dans ce cas, l'option "A" doit être spécifiée.

- Un programme sauvegardé doit pouvoir être rechargé en mémoire pour exécution.

Dans le DOS microsoft, plusieurs commandes réalisent cette fonction :

** LOAD

syntaxe LOAD <nom de fichier> [, ND [, R]]

Cette commande exécute un NEW de manière implicite (réinitialisation de la mémoire), puis charge en mémoire le programme spécifié, venant du disque 0 si ND est omis, et ferme tous les fichiers qui seraient restés ouverts lors des manipulations précédentes. Cependant, si l'option R est utilisée, ces fichiers restent ouverts et le programme chargé lancé automatiquement.

** RUN

syntaxe RUN <nom de fichier> [, ND [, R]]

Cette commande charge le programme spécifié et l'exécute automatiquement. Si l'option R est spécifiée, tous les fichiers restent ouverts, sinon ils sont automatiquement fermés. On peut noter que

RUN <nom de fichier> , ND , R

est équivalent à

LOAD <nom de fichier> , ND , R

** MERGE

syntaxe MERGE <nom de fichier> [, ND]

Contrairement aux deux précédentes, la commande MERGE permet de charger un programme sans effacer le contenu préalable de la mémoire : le programme chargé se rajoute au programme déjà résident. Si un numéro de ligne du programme chargé est identique à celui d'une ligne déjà en mémoire, la ligne du programme sur disque remplace celle qui était en mémoire. Cette commande permet d'effectuer de la segmentation dans les programmes BASIC.

Cette commande n'est utilisable que pour les fichiers contenant des programmes en ASCII (donc sauvegardés en utilisant la fonction "A" de SAVE)

Pratiquement aussi indispensables que les précédentes, on trouve encore deux fonctions :

- liste de la table des matières, ce qui permet de connaître l'ensemble des fichiers qui se trouvent sur une disquette.

syntaxe : FILES [ND[, ND]*]

- destruction d'un fichier (sous peine de voir le nombre de ceux-ci s'accroître et saturer la disquette).

syntaxe KILL <nom de fichier> [, ND]

Le fichier est supprimé de la directory et l'espace disque qu'il occupait réutilisable. Un fichier ouvert ne peut être détruit avant d'avoir été refermé.

D'autres fonctions sont très utiles pour manipuler les programmes (la syntaxe n'est donnée que pour les fonctions existant sur le DOS microsoft), cette liste n'étant pas limitative :

- recopie d'un programme, d'un fichier, d'une disquette
- protection de certains fichiers (en écriture, en lecture, contre toute destruction, ...)
- modification du nom d'un fichier

syntaxe : NAME <nom de fichier> AS <nom de fichier> [, ND]

- calcul de l'espace disponible sur une disquette

syntaxe : DSKF ND

- édition d'un fichier sur disque (on est obligé ici de charger le programme en mémoire puis de l'éditer : commandes LOAD puis LIST).

Exemple

Sur la disquette placée dans l'unité de disque 1, on dispose deux fichiers stockés sous forme compressée.

```
Programme "T1"      10   INPUT   A
                    20   PRINT   A
                    30   PRINT   "FINI"

Programme "T2"       5   A = 5
                    20   PRINT   A * A
                    40   PRINT   "OU PRESQUE"
```

Considérons alors la séquence d'ordres suivante :

```
MOUNT 1                    monter le disque 1 (nécessaire avant
                           tout travail sur ce disque)

FILES 1                    liste des fichiers existants

  T1        T2

LOAD "T1" , 1              chargement de T1

RUN                        }
                           }
 ? 3                       } exécution de T1
                           }
 3                         }
                           }
FINI                       }

RUN "T2", 1                }
                           } lancement de T2
25                         }
                           }
OU PRESQUE                 }

SAVE "T3", 1, A            sauvegarde en ASCII du programme en
                           mémoire dans T3

LOAD "T1", 1               chargement de T1

MERGE "T3", 1              rajout de T3 en mémoire
```

```
LIST

 5   A = 5
10   INPUT A
20   PRINT  A * A
30   PRINT "FINI"
40   PRINT "OU PRESQUE"
```

} Programme résultant

KILL "T1", 1 — destruction de T1

SAVE "T3", 1 — sauvegarde du programme dans T3

```
FILES 1

  T2          T3
```

UNLOAD 1 — recopie de la directory sur disque

IV 2 Les commandes d'accès aux fichiers

Pour des questions de simplicité, nous allons étudier séparément les accès aux fichiers de type séquentiel et aux fichiers à accès direct, les méthodes et les problèmes posés étant très différents.

IV 2 - 1 L'accès aux fichiers séquentiels

Dans un fichier séquentiel, les enregistrements peuvent être de longueur quelconque et différente d'un enregistrement à l'autre d' un même fichier. Ils sont rangés successivement sur le support. Cette longueur variable fait qu'il n'est pas possible d'accéder à un enregistrement sans accéder à tous ceux qui précèdent. D'autre part ce type de fichier ne peut être modifié : rien ne garantit que l'enregistrement que l'on veut écrire soit de même longueur que celui qui figurait au préalable. Aussi un fichier séquentiel ne pourra-til être accédé lors d'une manipulation que dans un mode à la fois :

- écriture (une seule fois, à sa création, il ne pourra être modifié par la suite).

- lecture (il pourra par contre être réexploré autant de fois qu'il sera nécessaire).

Nous avons vu que l'accès au fichier s'effectue à travers un canal. La première directive à utiliser, avant tout traitement du fichier, associera ce dernier à un des canaux disponibles. Il faudra "ouvrir" le fichier. Cet ordre précisera de plus le sens du transfert que l'on désire effectuer (entrée ou sortie) et l'unité de disque où se trouve le fichier

syntaxe : OPEN "mode" , NC, nom de fichier

OPEN <" mode"> ,# NC, <nom de fichier> [,ND]

Si ND n'est pas spécifié, il sera considéré par défaut comme nul (le fichier sera recherché ou créé sur le disque 0). NC indique le numéro de canal que l'on désire associer au fichier. Les modes d'ouverture sont :

- "O" si le fichier est utilisé en sortie (écriture). Le fichier est alors créé sur le disque indiqué. Si un fichier existe déjà sous ce nom, un message d'erreur sera édité.

- "I" si le fichier doit être utilisé en entrée (lecture). Il est possible d'ouvrir un même fichier sur plusieurs canaux en lecture.

Lorsque l'on a fini de travailler avec un fichier, il est nécessaire de restituer le canal (qui pourra ainsi être associé ultérieurement à un autre fichier), et de vider le buffer du canal. En effet, ce dernier peut contenir des informations dont le stockage sur le fichier a été demandé, et qui n'ont pas été écrites sur le support. Les commandes telles que NEW, LOAD, RUN, END referment automatiquement tous les canaux ouverts. D'autres commandes, telles KILL, NAME, ne peuvent être exécutées que sur des fichiers fermés.

syntaxe : CLOSE [# NC [,# NC]*]

Si NC n'est pas spécifié, tous les canaux ouverts sont refermés. Une fois qu'un fichier utilisé en sortie a été refermé, il ne peut être rouvert qu'en entrée.

Une fois qu'un fichier a été ouvert en sortie, pour écrire des informations, il suffit de donner un ordre d'édition sur le numéro de ca-

nal associé au fichier, en précisant la liste des variables à stocker.

syntaxe : PRINT # NC, <liste d'expressions>

Lorsque les informations sont destinées à être relues (la majorité des cas...), elles doivent être dans une forme acceptable pour l'ordre INPUT. Ainsi, les différentes variables d'un même enregistrement doivent être séparées par des virgules, les chaînes de caractères entre quotes (") si elles contiennent des virgules, ...
Par exemple la ligne :

```
100  PRINT # 1, A ; B ; C
```

produira un enregistrement qui ne pourra être relu. Il faudrait écrire :

```
100  PRINT # 1, A ; " ; " ; B ; " , " ; C
```

qui produira un enregistrement correct.

La lecture d'un enregistrement sur un fichier ouvert en entrée s'effectue à l'aide d'une commande semblable à la demande d'une entrée au clavier.

syntaxe INPUT # NC, <liste de variable>

La liste de variable doit être satisfaite par les informations contenues dans l'enregistrement (le nombre de données doit correspondre, ainsi que leur nature).

Ainsi un enregistrement créé par l'instruction précédente pourra être lu par (à condition que les associations fichier<->canal soient correctement effectuées).

```
200    INPUT # 2, A1 , A2, A3
```

Lors de l'exploration d'un fichier séquentiel, il est possible de tester si le dernier enregistrement lu est le dernier du fichier ou pas.

syntaxe X = EOF (NC)

Si l'on a atteint la fin du fichier associé au canal NC,
alors : EOF (NC) = - 1
sinon : EOF (NC) = 0

Exemple

Considérons le cas d'un annuaire simplifié. On dispose d'une liste alphabétique de personnes d'un même département, et on désire exploiter cet annuaire de manière automatique. On supposera que l'on ne pourra pas rajouter de nom dans l'annuaire, mais qu'une personne peut voir son numéro de téléphone changer. On pourra alors utiliser les programmes suivants :

* **Création du fichier**

```
10  OPEN "O" ,#1,  "ANNUAIRE" ------ ouverture du fichier où
                                      sera rangé l'annuaire
20  INPUT "NOM", A$ ----------------- entrée du nom de la per-
                                      sonne
30  INPUT "TEL", C------------------- et du numéro de téléphone
                                      (numéro à 6 chiffres)
40  PRINT# 1, A$  ; " , " ; C------- stockage des informations
50  INPUT "AUTRE NOM" , Y$ ---------- annuaire terminé ?
60  IF Y$  = "OUI" THEN  GO TO 20---- annuaire terminé ?
70  CLOSE#1                           oui  : fermer le fichier
80  END
```

* exploration du fichier

```
10  OPEN "I", # 2, "ANNUAIRE"        ouverture du fichier
20  INPUT "NOM", A$                  nom recherché
30  INPUT# 2, B$ , C                 lecture d'un enregistrement
40  IF A$  = B$  THEN GO TO 80       est-ce le nom recherché ?
50  IF EOF (2) = 0 THEN GO TO 30     non ; a-t-on fini d'explo-
                                     rer le fichier ?
60  PRINT "NOM INEXISTANT"           oui, nom non trouvé
70  GO TO 90
80  PRINT "TELEPHONE :" ; C          nom trouvé ; édition du tel.
90  CLOSE # 2                        exploration terminée ; ferme-
                                     ture du fichier
100 INPUT "AUTRE NOM", Y$            autre nom recherché  ?
110 IF Y$ = "OUI" THEN GO TO 10      oui ; retour à l'ouverture du fichi
120 END                              non ; arrêt du programme
```

* modification

```
10  OPEN "I",#1, "ANNUAIRE"          ouverture du fichier à modifier
20  OPEN "O",#2, "TEMPO"             ouverture fichier temporaire
                                     (stockage de l'annuaire modifié)
```

```
30  INPUT  "NOM", A$                    nom recherché
40  INPUT#1, B$ , C                     lecture enregistrement
50  IF A $ = B$  THEN GO TO 110         nom recherché ?
60  PRINT#2, B$  ; ", " ; C             non ; recopie de l'enregistrement
                                        sur TEMPO.
70  IF EOF (1) = 0 THEN GO TO 40        fin de ANNUAIRE ?
80  PRINT  "NOM INEXISTANT"             oui ; nom pas touvé
90  CLOSE#1, #2                         fermeture des fichiers
100 KILL "TEMPO"                        destruction TEMPO, inutile la
                                        modification n'ayant eu lieu.
110 INPUT "NUM. TEL." , C               nom trouvé ; nouveau numéro
                                        de téléphone ?
120 PRINT #2, B$  ; ","; C              stockage de l'enregistrement
                                        modifié sur TEMPO.
130 IF EOF (1) = 0 THEN GO TO 170       fin de annuaire ?
140 INPUT#1, B$ , C                     non ; recopie de la fin de
                                        ANNUAIRE su TEMPO
150 PRINT#2, B$  ; "," ; C
160 GO TO 130
170 CLOSE#1,#2                          oui ; fermeture des fichiers.
180 KILL "ANNUAIRE"                     remplacement de ANNUAIRE par
                                        fichier modifié
190 NAME "TEMPO" AS "ANNUAIRE"
200 INPUT "AUTRE MODIF.", Y$            autre mise à jour ?
210 IF Y$ = OUI" THEN GO TO 10          oui ; retour à Y l'ouverture des
                                        fichiers.
220 END                                 non ; fin du programme
```

Deux remarques s'imposent :

- pour une recherche, il peut être nécessaire de lire tout le fichier avant de trouver le nom demandé (si l'on a N personnes dans l'annuaire, le temps moyen d'accès correspondra à la lecture de N/2 enregistrements)
- pour une mise à jour, il est nécessaire de recopier le fichier dans sa totalité pour chaque modification (en fait, une seule recopie est nécessaire si on ordonne les modifications par ordre alphabétique).

IV 2 - 2 Les fichiers à accès direct

L'accès direct, au prix d'un logiciel plus complexe à mettre en oeuvre, permettra par contre d'accélérer grandement les opérations de recherche et de mise à jour. On pourra ici accéder à

un enregistrement par son numéro d'ordre. Ce numéro sera dit logique. Cela signifie qu'il ne correspond pas à la place physique de l'enregistrement sur le support, mais au numéro relatif de l'enregistrement par rapport au début du fichier. C'est le D.O.S. qui gère automatiquement le passage de ce numéro logique à l'emplacement physique.

Nous avons signalé que ce type d'accès est plus rapide. En contrepartie, un certain nombre d'opérations qui étaient transparentes à l'utilisateur pour les fichiers séquentiels (c'est-à-dire que l'utilisateur n'avait pas à les connaître) devront être demandées de manière explicite. Il en est ainsi des transferts physiques entre le buffet du canal de transmission et le disque, du test de fin de fichier, de l'indication du format des données dans l'enregistrement, ...

Comme dans le cas des fichiers séquentiels, aucune opération ne pourra être effectué sur un fichier avant qu'il ait été ouvert.

syntaxe : OPEN "R" ,#NC, <nom de fichier> [, ND]

La seule différence avec l'accès séquentiel est dans l'indication du mode d'ouverture ("R" , au lieu de "I" ou "O"). Il faut noter cependant qu'un fichier créé en accès séquentiel ne peut être exploré en accès direct et réciproquement. Par contre un tel fichier est ouvert simultanément en lecture et en écriture. Si le fichier n'existait pas, il se trouve créé par l'ordre OPEN.

De même qu'un fichier séquentiel, un fichier en accès direct doit être refermé une fois les accès disque terminés, afin de remettre à jour les descripteurs du fichier et de restituer le canal, et la syntaxe est la même :

CLOSE [#NC] [, #NC *]

Les ordres de transfert sont à gérer par l'utilisateur, qui dispose à cet effet de deux commandes :

- transfert d'un enregistrement du disque vers le buffer du canal.

syntaxe : GET #NC [, NE]

où NE représente un numéro d'enregistrement. Cet ordre transfere le contenu du secteur logique NE du fichier dans le buffer. L'unité d'enregistrement correspond à la longueur d'un secteur : ici 128 octets. Si NE n'est pas spécifié, c'est l'enregistrement suivant le dernier transféré qui sera lu (lecture du 1er enregistrement s'il s'agit du premier accès). Aucun test n'est effectué quand au dépassement du dernier enregistrement écrit.

- transfert du contenu du buffet du canal vers le disque.

syntaxe : PUT # NC [, NE]

Il s'agit là de l'opération inverse de la précédente, les mêmes remarques étant valables.

Afin de se protéger contre des accès en dehors du fichier, l'utilisateur dispose de deux fonctions. La première permet de connaître le numéro du dernier enregistrement accédé plus un, plus précisément le numéro de l'enregistrement qui était accédé par PUT ou GET si NE n'est pas spécifié.

syntaxe : X = LOC (NC)

L'autre fonction permet de connaître le numéro d'enregistrement le plus élevé du fichier, autrement dit le dernier numéro d'enregistrement écrit sur le fichier.

syntaxe : X = LOF (NC)

On devra normalement toujours avoir

LOC (NC) $\leq$ LOF (NC) + 1

Un enregistrement une fois transféré dans le buffer, il est nécessaire de préciser quel est le format des informations qu'il contient pour pouvoir y accéder. Ces informations ne peuvent être que des caractères ASCII.

syntaxe : FIELD #NC, <NL> < AS ><VAR$>[,<NL>< AS ><VAR$>]*

'enregistrement se trouve ainsi fragmenté en champs, chaque champs correspondant à une variable du type chaîne de caractères, NL précisant la longueur de la chaîne de caractères.

FIELD #1, 10 AS A1$, 8 AS A2$, 12 ASB$

affecte les 10 premiers octets du buffet à la variable AI$, les 8 suivants à A2$, les 12 suivants à B$. La somme des NL devra rester inférieure à la longueur de l'enregistrement (au maximum 128 octets).

Cette correspondance étant établie, les variables chaîne de caractères peuvent intervenir dans toute expression compatible avec leur type. Par contre elles ne peuvent se voir affecter une valeur que par l'intermédiaire de fonctions particulières : LSET et RSET.

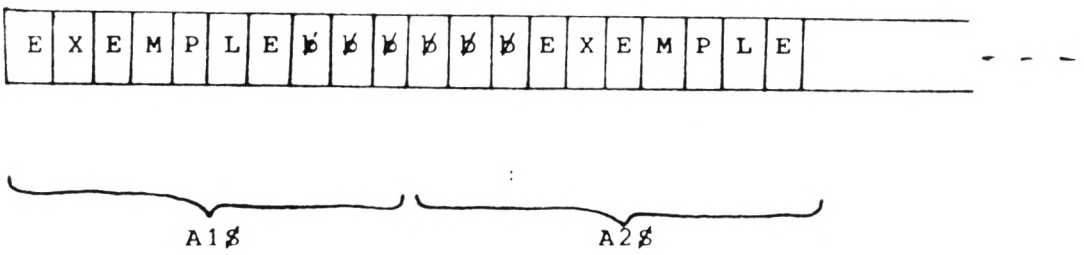
syntaxe : LSET (var$) = ⟨expression de concatenation⟩
RSET (var$) = ⟨expression de concatenation⟩

Dans les deux cas, le résultat de l'expression est rangé dans la partie du buffer pointée par Var$. Ces fonctions ont un comportement différent si le résultat de l'expression est une chaîne de longueur inférieure à la zone de buffer affectée à Var$. Dans les deux cas la chaîne est complétée par des "espace", mais après la chaîne avec LSET, avant la chaîne avec RSET.

Exemple

```
10    FIELD #1, 10 ASAI$,   10 ASA2$
20    A$ = "EXEMPLE"
30    LSET  (AI$) = A$
40    RSET  (A2$) = A$
```

Après éxécution, le buffer du canal 1 contiendra :



Nous avons vu que seules des chaînes de caractères peuvent figurer dans l'enregistrement. Aussi dispose-t-on de fonctions de conversion permettant le passage d'une variable numérique à une chaîne de caractè-

res et réciproquement.

- Conversion numérique —> chaîne de caractères

  * MKI$ (valeur entière). convertit une valeur numérique entière en une chaîne de caractères de 2 octets.

  * MKS$ (valeur simple précision) convertit une valeur numérique simple précision en une chaîne de caractères de 4 octets.

  * MKD$ (valeur double précision) convertit une valeur numérique double précision en une chaîne de caractères de 8 octets.

- Conversion chaîne de caractères->numérique

  * CVI (chaîne de 2 octets) convertit la chaîne en un nombre entier.

  * CVS (chaîne de 4 octets) convertit la chaîne en un nombre simple précision.

  * CVD (chaîne de 8 octets) convertit la chaîne en un nombre double précision.

Exemple

Reprenons l'exemple de l'annuaire, traité avec les fichiers séquentiels.

* création

```
10  OPEN "R",#1, "TELEPH"        ouverture et création du fichier
20  FIELD#1, 8 ASN$, 8 ASNT$     format des données
30  INPUT "NOM", N1$             } entrée du nom
40  INPUT "NUM.TEL.", N2$        }      et du numéro de téléphone
50  LSET (N$) = N1$              } rangement dans N$
60  LSET (NT$) = N2$             }      et NT$
70  PUT#1                        stockage de l'enregistrement
80  INPUT "AUTRE NOM", Y$        autre nom ?
```

```
 90  IF Y$ = "OUI" THEN GO TO 30    oui ; entrée de l'autre nom
100  CLOSE #1                       non ; création terminée.
110  END
```

* Exploration

Elle pourait se faire comme pour l'accès séquentiel, en recherchant successivement dans chaque enregistrement si il correspond au nom recherché. Nous allons plutôt profiter de l'acces direct et du fait que les noms sont classés par ordre alphabétique pour utiliser une méthode dite de dichotomie. Cette méthode consiste à rechercher dans quelle moitié du fichier se trouve le nom demandé en comparant ce dernier au nom contenu dans l'enregistrement du milieu du fichier : moitié haute s'il est supérieur, moitié basse sinon. Puis on redivise la moitié de fichier ainsi sélectionnée en 2 par la même méthode...

* Principe

```
BINF = 1
BMAX = N

tant que      BMAX ≥ BINF

        Faire  BMILIEU = (BINF + BMAX)/2
               lire (BMILIEU)
               si   (BMILIEU) = résultat cherché alors  trouvé
               si   (BMILIEU) < résultat cherché alors  BINF = BMILIEU+1
                                                  sinon  BMAX = BMILIEU-1

        fait

faire     éditer  "N'existe pas"

fin

trouver, faire éditer  "numéro de téléphone"
```

* Programme

```
10 OPEN "R" ,#1, "TELEPH"        ouverture du fichier
```

```
20 FIELD#1, 8ASNOM$, OASNT$          définition des champs
                                     (la même que lors de la création)
30 INPUT "NOM", , N$                 nom recherché
40 BINF = 1                          }bornes de la recherche
50 BMAX = LOF (1)                    }
60 BM = (BMAX + BINF)/2
70 GET#1, BM
80 IF NOM$ = N$ THEN GO TO 140       trouvé
90 IF NOM$ > N$ THEN BMAX = BM -1
100 IF NOM$ < N$ THEN BINF = BM + 1
110 IF NOT (BMAX<BINF) THEN GO TO 60
120 PRINT  "N'EXISTE PAS"
130 GO TO 150
140 PRINT  "NUM.TEL. =" ; NT$
150 INPUT "AUTRE NOM", Y$            }autre recherche ?
160 IF Y$ = "OUI" THEN GO TO 30      }
170 CLOSE #1                         fermeture du fichier
180 END
```

* Mise à jour

Ce problème se ramène pratiquement au précédent. En effet un fichier en accès direct est ouvert en lecture et écriture. De ce fait, il n'y a pas nécessité de fichier temporaire. Il suffit, une fois le nom trouvé, de modifier le numéro de téléphone et de réécrire l'enregistrement modifié, soit les modifications suivantes au programme.

```
140 INPUT "NUM.TEL.", N1$            entrée du nouveau numéro
142 LSET (NT$) = N1$                 écriture dans le buffet
144 PUT#1, LOC (1) - 1               écriture dans le fichier
```

Remarques

- Ici, le nom doit être de longueur limitée. On est obligé de prévoir la longueur maximale dans la définition du champ à la création (restriction qui n'existait pas en accès séquentiel)

- la longueur du champ attribué au numéro de téléphone permet le stockage de l'indicatif départemental (mémorisation d'un numéro à 8 chiffres)

- l'espace disque est nettement sous-occupé. En effet, dans chaque enregistrement de 128 octets, seuls les 16 premiers sont occupés.

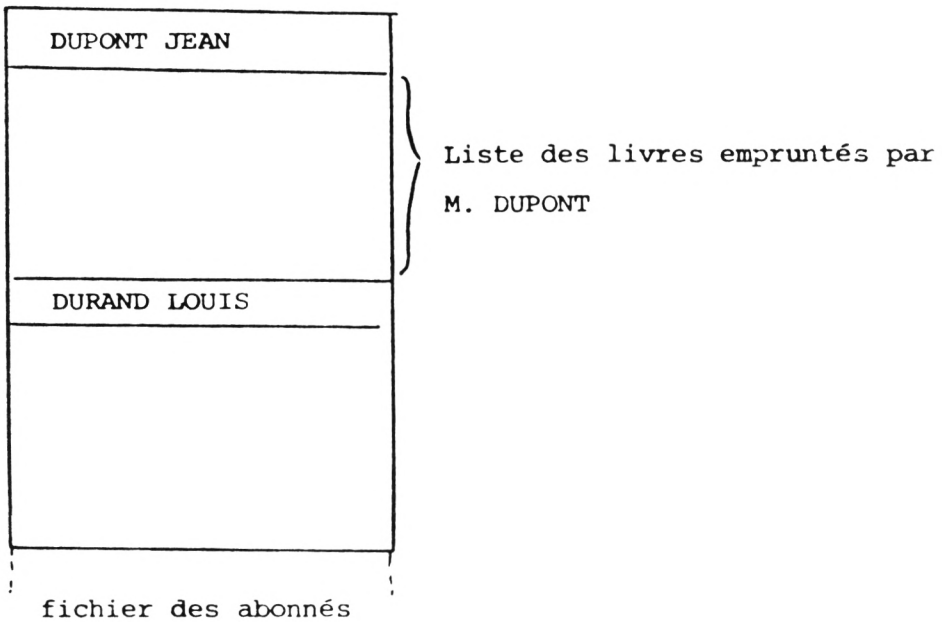
Il serait possible en fait de stocker 8 noms et numéros de téléphone en créant des sous-enregistrements (ceci peut se faire avec l'ordre FIELD).

- Les accés ont été considérablement accélérés. Si on a 1000 personnes dans le fichier, l'organisation en séquentiel imposerait environ 500 accès à chaque recherche (ce nombre pouvant monter à 1000). Avec l'accès direct et la méthode de recherche utilisée, le nombre maximum d'accès tombe à 10, avec une moyenne de l'ordre de 9,2 accès par recherche.

- Par contre, si l'on recherche un autre renseignement que le nom (par exemple l'indicatif départemental), on sera à nouveau obligé de faire une recherche séquentielle. L'accès direct ne facilite la recherche que vis-à-vis du critère selon lequel le fichier a été classé. Pour accélérer la recherche des indicatifs, il serait nécessaire de créer un autre fichier contenant pour chaque indicatif possible des pointeurs vers le fichier des noms.

V NOTION DE BASE DE DONNEES - SYSTEME DE GESTION DE BASE DE DONNEES

Si nous avons dans une bibliothèque à gérer le prêt des livres, il est possible à chaque emprunt d'un ouvrage de l'inscrire dans le fichier des abonnés à la bibliothèque.



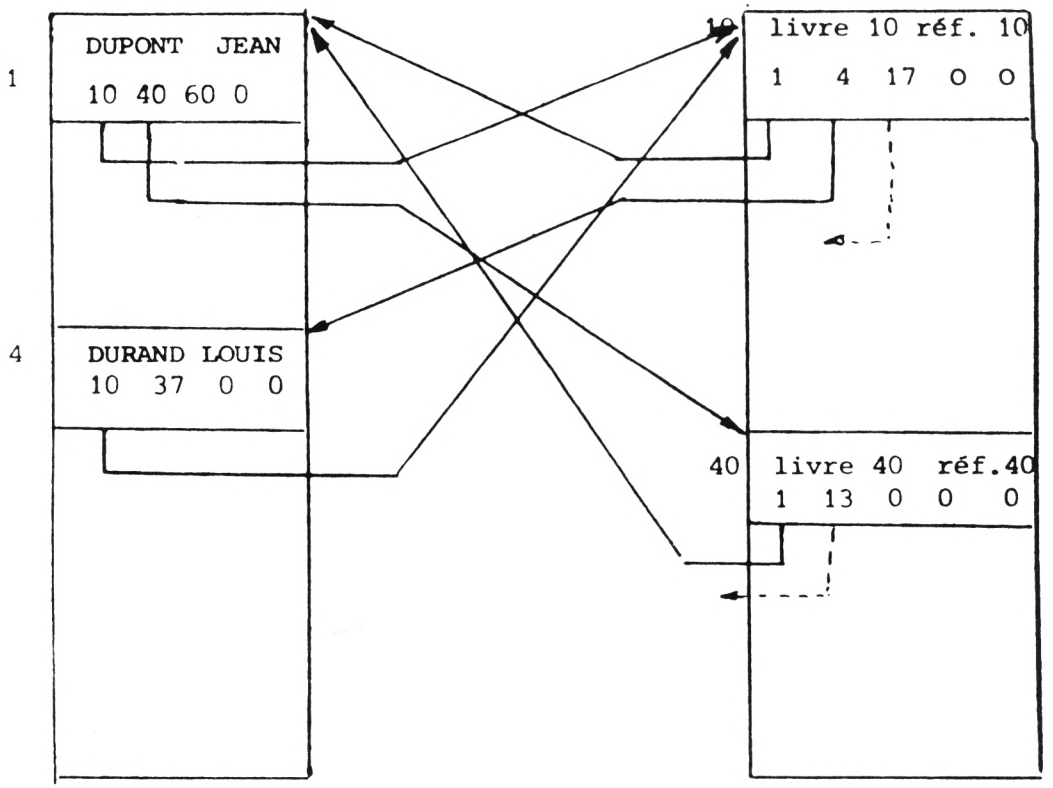
Il n'est pas possible de savoir immédiatement avec cette structure où se trouve un exemplaire d'un livre. En outre les renseignements concernant un ouvrage seront dupliqués chaque fois qu'un exemplaire supplémentaire sera prêté.

Il est possible de rendre indépendantes les données en éclatant ce fichier en un fichier des abonnés et un fichier des livres.

Les fichiers seront liés de la manière suivante :

- chaque abonné pointera sur l'ensemble des"livres" qu'il a emprunté.
- un enregistrement du fichier "livres" contiendra le nom et les références du livre.

Le nombre d'exemplaires existant à la bibliothèque et une série de pointeurs vers les personnes ayant emprunté un exemplaire du livre.



fichiers des abonnés

fichier des livres

Cette nouvelle structure nous permet aisément de connaître :

- pour chaque abonné, la liste des livres qu'il a emprunté,
- pour chaque livre, le nombre d'exemplaires empruntés et le nom des emprunteurs.

Les deux fichiers utilisés indépendamment permettent d'obtenir

- la liste des abonnés
- la liste des ouvrages de la bibliothèque.

Cette structure est de type base de données. On peut définir une base de données comme étant :

"un ensemble de données reliées agissant comme support des besoins de l'organisation et dont les caractéristiques sont :

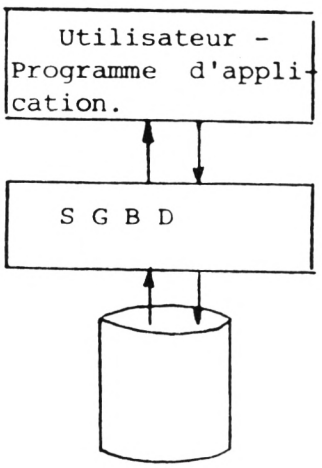
a) l'indépendance de la structure des données par rapport à la structure des traitements,
b) la possibilité de prendre en compte des relations des différentes données entres elles.

c) la non redondance sémantiques des données,

d) le partage des données entre les programmes d'application et les interrogations ad hod".

[Référence 6]

L'accés aux données dans une telle structure est rapide mais complexe du point de vue programmation. Aussi la plupart des constructeurs de matériel informatiques porposent-ils des logiciels permettant de gérer automatiquement les données. Il s'agit des systèmes de gestion de base de données (SGBD). Un SGBD est un ensemble de procédures réalisant l'interface entre l'utilisateur et la base de données.



Ces procédures permettent de créer la base en décrivant les différentes clés d'accès souhaitées et de faciliter l'accès aux données de part la structuration introduite.

Il est à noter que ces systèmes, qui sont utilisés depuis longtemps dans les gros ensembles de calculs, sont encore à l'état embryonnaires sur les micro-machines.

Bibliographie

- LE BASIC

1 - PIERRE LEBEUX : Introduction au BASIC
Editions SYBEX

2 - R.L. ALBRECHT : BASIC A self teaching guide
John Wiley and sons, Inc

- LA MICROINFORMATIQUE

3 - Rodnay ZAKS : Pierre LE BEUX Les microprocesseurs
Editions SYBEX

4 - M. AUMIAUX : L'emploi des microprocesseurs
MASSON

5 - Rodnay ZAKS : Introduction aux micro-ordinateurs à usage individuel et professionnel
Editions SYBEX

- LES BASES DE DONNEES

6 - M. HABRIAS : Les bases de données
convention d' étude 78.498. CFPC

- REVUES

7 - l'ordinateur individuel

8 - Micro-systèmes

9 - BYTE

10 - Interface Age