Les fichiers

Christian Lasou, Nour-Eddine Oussous, Éric Wegrzynowski

Licence ST-A, USTL - API1

19 mars 2007

Caractéristiques des fichiers

Utiles pour conserver toute sorte d'informations (programmes, textes, données, ...), les fichiers possèdent certaines caractéristiques.

Texte ou binaire ?

Certains fichiers contiennent du texte

```
[eric@Okocim:Fichiers]$ cat exemple1.pas
// auteur : API1
// date   : mars 2007
// objet  : programme exemple de
//          creation d'un fichier type
...
```

Texte ou binaire ?

Certains fichiers contiennent du texte

```
[eric@Okocim:Fichiers]$ cat exemple1.pas
// auteur : API1
// date   : mars 2007
// objet  : programme exemple de
//          creation d'un fichier type
...
```

d'autres non

```
[eric@Okocim:Fichiers]$ cat exemple1.o
ELFI4(                  U\unhbox \voidb@x \bgroup \let \unhbox \voidb@x \setbox \@tempboxa \
L\unhbox \voidb@x \bgroup \accent 127o\penalty \@M \hskip \z@skip \egroup \unhbox \voidb@x
...
```

Texte ou binaire ?

Certains fichiers contiennent du texte

```
[eric@Okocim:Fichiers]$ cat exemple1.pas
// auteur : API1
// date   : mars 2007
// objet  : programme exemple de
//          creation d'un fichier type
...
```

d'autres non

```
[eric@Okocim:Fichiers]$ cat exemple1.o
ELFI4(                  U\unhbox \voidb@x \bgroup \let \unhbox \voidb@x \setbox \@tempboxa \
L\unhbox \voidb@x \bgroup \accent 127o\penalty \@M \hskip \z@skip \egroup \unhbox \voidb@x
...
```

Les premiers sont des fichiers de texte, les seconds des fichiers binaires. Seuls les premiers sont éditables à l'aide d'un éditeur de textes. L'exploitation des autres se fait avec des programmes appropriés à chaque situation.

Fichiers typés

Fichier typé =

Fichiers typés

Fichier typé =

- fichier binaire

Fichiers typés

Fichier typé =

- fichier binaire
- découpables en articles (ou enregistrements)

Fichiers typés

Fichier typé =

- fichier binaire
- découpables en articles (ou enregistrements)
- tous de même type.

Fichiers typés

Fichier typé =

- fichier binaire
- découpables en articles (ou enregistrements)
- tous de même type.

Remarque : tout fichier peut être considéré comme un fichier typé dont chaque article est un octet.

Déclaration des fichiers en Pascal

En Pascal, on déclare un fichier typé par **FILE of** <**type**> où <**type**> est le type des articles du fichier.

Exemple

```
var
   f1 : FILE of CHAR;
   f2 : FILE of CARDINAL;
   f3 : FILE of STRING[10];
   f4 : FILE of DATE;
```

Exemple

On peut nommer un type fichier.

Exemple

Déclaration d'un type « fichier de points » et d'une variable de ce type.

```
type
    POINT = record
        x,y : REAL;
    end {POINT};

    FICHIER_POINTS = file of POINT;

var
    fp : FICHIER_POINTS;
```

Fichier logique et fichier physique

- fichier physique = fichier stocké sur un périphérique (CDROM, disque dur, ...). Il possède un nom. Il peut préexister et/ou perdurer aux programmes qui le manipulent.

Fichier logique et fichier physique

- fichier physique = fichier stocké sur un périphérique (CDROM, disque dur, ...). Il possède un nom. Il peut préexister et/ou perdurer aux programmes qui le manipulent.
- fichier logique = fichier vu du programme qui le manipule. C'est une variable d'un type fichier désignée par un identificateur. Sa durée de vie égale celle du programme qui le contient.

Fichier logique et fichier physique

- fichier physique = fichier stocké sur un périphérique (CDROM, disque dur, ...). Il possède un nom. Il peut préexister et/ou perdurer aux programmes qui le manipulent.
- fichier logique = fichier vu du programme qui le manipule. C'est une variable d'un type fichier désignée par un identificateur. Sa durée de vie égale celle du programme qui le contient.

$\Rightarrow$ Nécessité d'associer un fichier physique à tout fichier logique : c'est l'assignation.

L'assignation

assign

```
// associe le fichier physique désigné par nom
// au fichier logique f
procedure assign(var f : <fichier>;
                 const nom : STRING);
```

Le paramètre nom est une chaîne désignant l'emplacement et le nom du fichier (la syntaxe peut dépendre du système d'exploitation). L'emplacement peut être relatif ou absolu.

Exemples

Fichier physique dans le répertoire courant

```
assign(fp,'sinus.data');
```

Exemples

Fichier physique dans le répertoire courant

```
assign(fp,'sinus.data');
```

Fichier physique donné par un chemin absolu

Version Unix

```
assign(fp,'/home/calbuth/API1/sinus.data');
```

Exemples

Fichier physique dans le répertoire courant

```
assign(fp,'sinus.data');
```

Fichier physique donné par un chemin absolu

Version Unix

```
assign(fp,'/home/calbuth/API1/sinus.data');
```

Fichier physique donné par un chemin absolu

Version Windows

```
assign(fp,'C:\API1\sinus.data');
```

Différentes ouvertures

On peut distinguer plusieurs ouvertures d'un fichier

Différentes ouvertures

On peut distinguer plusieurs ouvertures d'un fichier

- l'ouverture en création : c'est le mode d'ouverture qu'il faut choisir lorsqu'on veut créer un nouveau fichier (physique).

Différentes ouvertures

On peut distinguer plusieurs ouvertures d'un fichier

- l'ouverture en création : c'est le mode d'ouverture qu'il faut choisir lorsqu'on veut créer un nouveau fichier (physique).
- l'ouverture en consultation et modification : c'est le mode d'ouverture qu'il faut choisir lorsqu'on veut
  - lire des informations contenues dans un fichier,
  - et/ou ajouter ou modifier des informations dans ce fichier.

  Le fichier physique doit exister préalablement.

Ouverture en création

rewrite

```
// ouvre le fichier f en création
// le fichier physique associé est vide
procedure rewrite(var f : <fichier>);
```

Remarque : si le fichier physique associé à f existe, il est vidé de son contenu.

Ouverture en consultation/modification

reset

```
// ouvre le fichier f en consultation
// ou modification
// CU : le fichier physique doit exister
procedure reset(var f : <fichier>);
```

Fermeture d'un fichier

close

```
// ferme le fichier f
procedure close(var f : <type fichier>);
```

Remarque : après fermeture aucune opération sur le fichier n'est possible, hormis une ouverture.

Exemple

Exemple (non complet)

```
1  program exemple1;
2
3  type
4     POINT = record
5        x,y : REAL;
6     end {POINT};
7
8     FICHIER_POINTS = file of POINT;
9
10 var
11    fp : FICHIER_POINTS;
12
13 begin
14    assign(fp,'sinus.data');
15    rewrite(fp);
16
17    {traitement du fichier}
18
19    close(fp);
20 end.
```

Écriture d'un article

write

```
procedure write(var f : <fichier>;
                const a : <article>);
```

Écriture d'un article

write

```
procedure write(var f : <fichier>;
                const a : <article>);
```

1. Avant écriture, le fichier doit être ouvert ;

Écriture d'un article

write

```
procedure write(var f : <fichier>;
                const a : <article>);
```

1. Avant écriture, le fichier doit être ouvert ;
2. le pointeur peut pointer vers la fin de fichier : c'est un ajout d'article ;

Écriture d'un article

write

```
procedure write(var f : <fichier>;
                const a : <article>);
```

1. Avant écriture, le fichier doit être ouvert ;
2. le pointeur peut pointer vers la fin de fichier : c'est un ajout d'article ;
3. le pointeur peut pointer vers un article existant : c'est une modification d'article ;

Écriture d'un article

write

```
procedure write(var f : <fichier>;
                const a : <article>);
```

1. Avant écriture, le fichier doit être ouvert ;
2. le pointeur peut pointer vers la fin de fichier : c'est un ajout d'article ;
3. le pointeur peut pointer vers un article existant : c'est une modification d'article ;
4. après écriture, le pointeur pointe sur l'article suivant.

Ajout d'un article

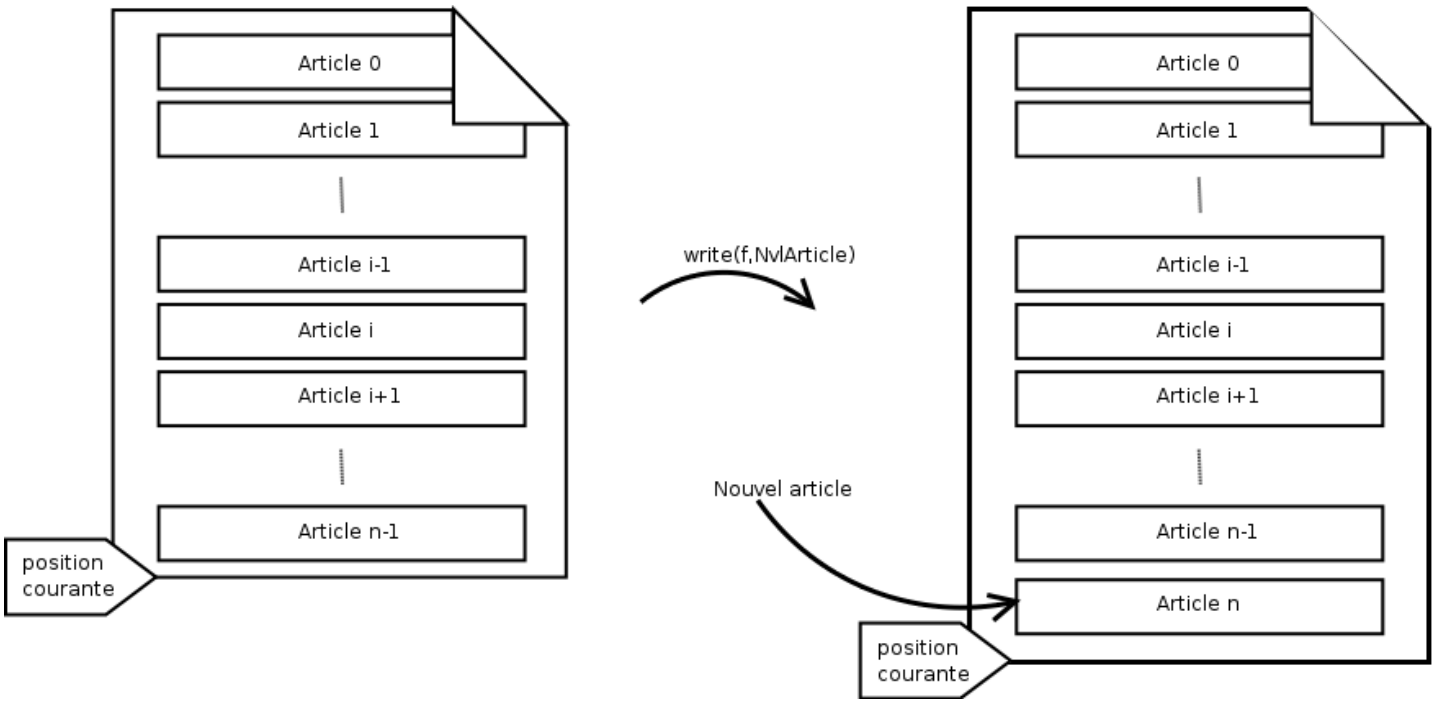

FIG.: Ajout d'un article en fin de fichier

Exemple

Création d'un fichier de points

```
5  program exemple1;
6
7  type
8     POINT = record
9        x,y : REAL;
10    end {POINT};
11
12    FICHIER_POINTS = file of POINT;
13
14 var
15    fp : FICHIER_POINTS;
16    i : CARDINAL;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    rewrite(fp);
21    for i := 0 to 99 do begin
22       p.x := 2*i*pi/100;
23       p.y := sin(p.x);
24       write(fp,p);
25    end {for};
26    close(fp);
27 end.
```

Lecture d'un article

read

```
procedure read(var f : <fichier>;
               out a : <article>);
```

Lecture d'un article

read

```
procedure read(var f : <fichier>;
               out a : <article>);
```

1. Avant lecture, le fichier doit être ouvert ;

Lecture d'un article

read

```
procedure read(var f : <fichier>;
               out a : <article>);
```

1. Avant lecture, le fichier doit être ouvert ;
2. le pointeur ne doit pas pointer vers la fin de fichier ;

Lecture d'un article

read

```
procedure read(var f : <fichier>;
               out a : <article>);
```

1. Avant lecture, le fichier doit être ouvert ;
2. le pointeur ne doit pas pointer vers la fin de fichier ;
3. l'article lu est l'article numéro i ;

Lecture d'un article

read

```
procedure read(var f : <fichier>;
               out a : <article>);
```

1. Avant lecture, le fichier doit être ouvert ;
2. le pointeur ne doit pas pointer vers la fin de fichier ;
3. l'article lu est l'article numéro i ;
4. après lecture, le pointeur pointe sur l'article suivant.

Lecture d'un article

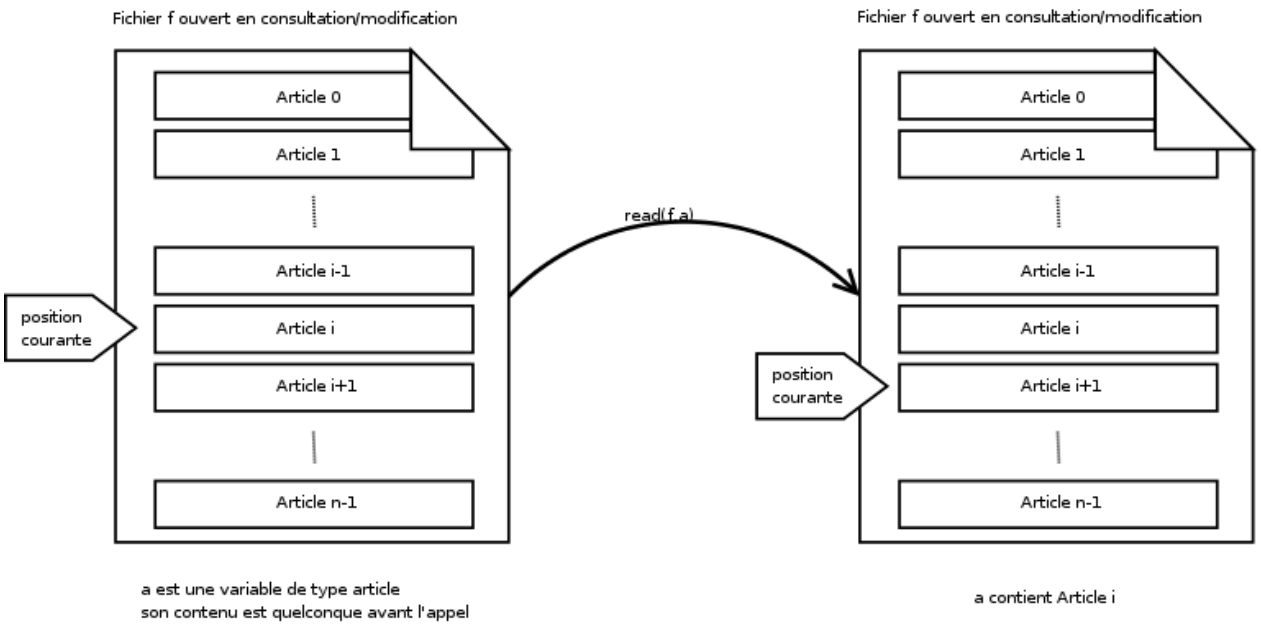

FIG.: La procédure **read**

La fin de fichier

eof

```
// détermine si le pointeur est
// en fin du fichier f
function eof(const f : <fichier>) : BOOLEAN;
```

La fin de fichier

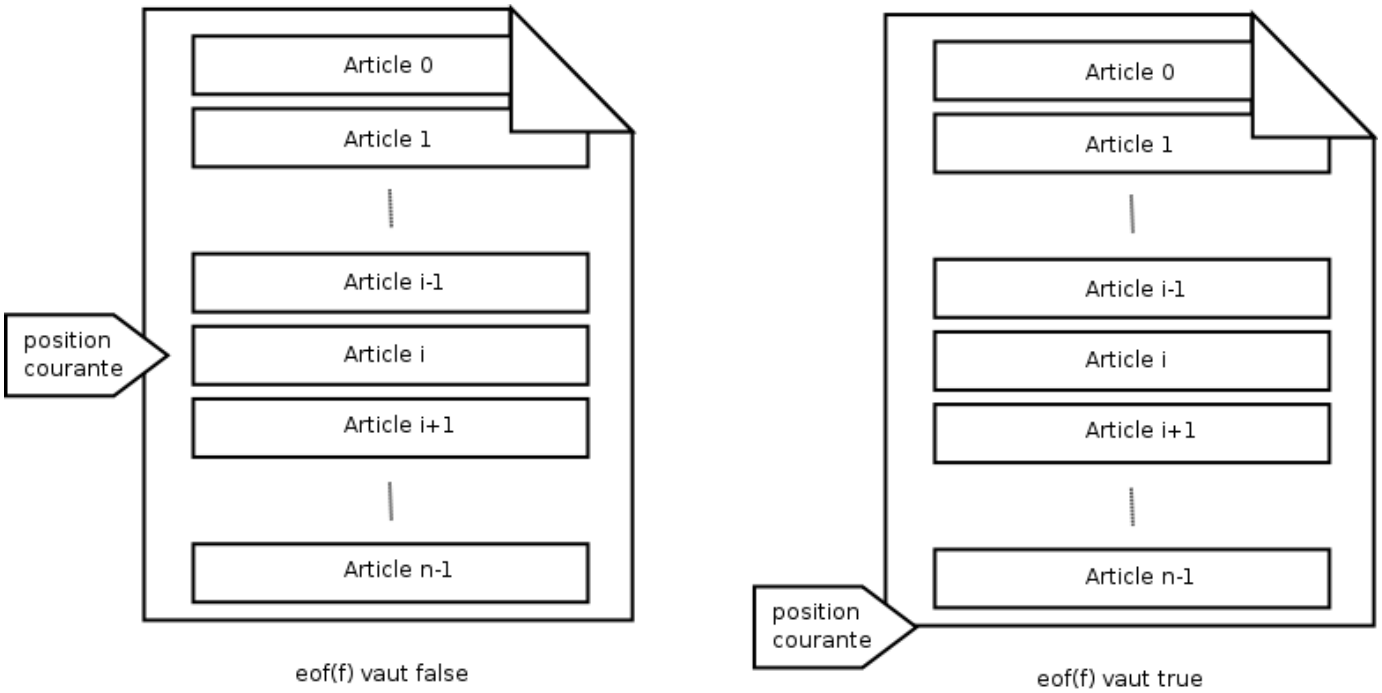

FIG.: La fonction **eof**

Exemple

Lecture et affichage des articles d'un fichier

```
5  program exemple2;
6
7  type
8     POINT = record
9        x,y : REAL;
10    end {POINT};
11
12    FICHIER_POINTS = file of POINT;
13
14 var
15    fp : FICHIER_POINTS;
16    i : CARDINAL;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    i := 0;
22    while not eof(fp) do begin
23       read(fp,p);
24       writeln(i:3,p.x:7:3,p.y:7:3);
25       inc(i);
26    end {for};
27    close(fp);
28    writeln(i,' enregistrements lus');
29 end.
```

Numérotation

- Dans un fichier typé, les articles (ou fiches, ou enregistrements) sont numérotés à partir par les entiers successifs de 0 à $n-1$, où n désigne le nombre d'articles contenus dans le fichier.

Numérotation

- Dans un fichier typé, les articles (ou fiches, ou enregistrements) sont numérotés à partir par les entiers successifs de 0 à $n-1$, où n désigne le nombre d'articles contenus dans le fichier.
- À tout moment, un fichier logique pointe
  - soit vers un article qui pourra être lu ou modifié,
  - soit vers une position qui suit le dernier article, qui pourra accueillir le prochain enregistrement (ajout d'un article).

Numérotation

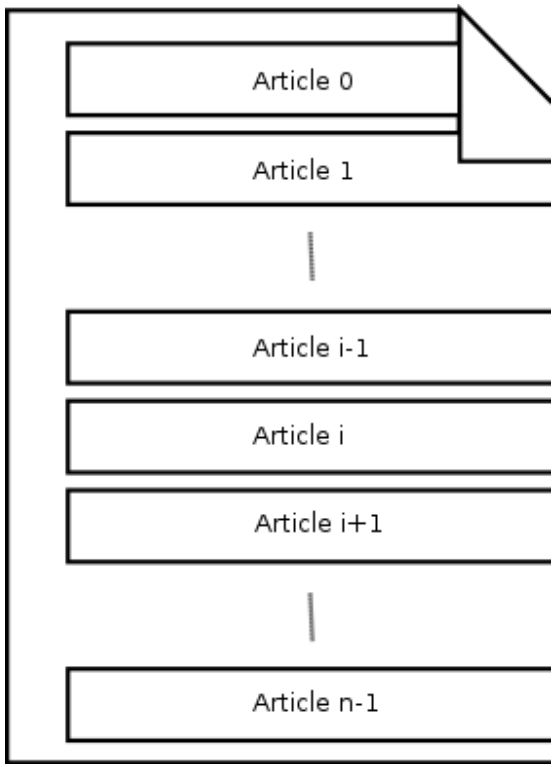

FIG.: Un fichier de *n* articles

Nombre d'articles

filesize

```
// donne le nombre d'articles dans le fichier f
function filesize(const f :<fichier>): CARDINAL;
```

Nombre d'articles

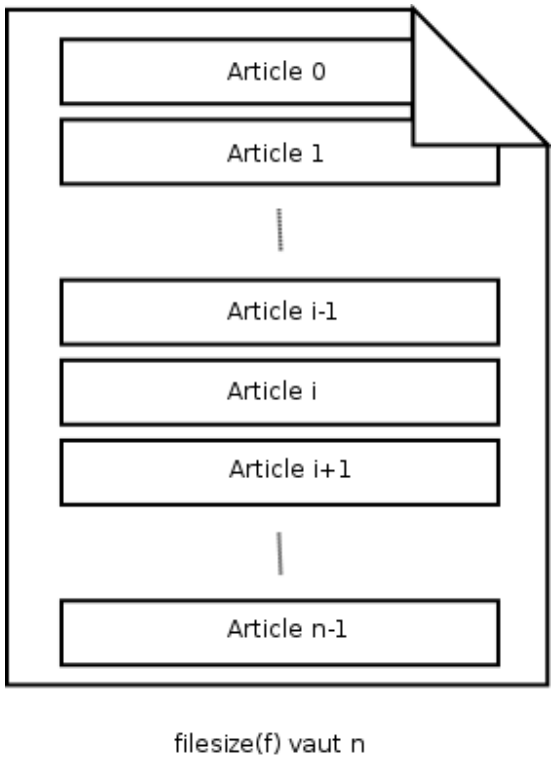

FIG.: La fonction **filesize**

Position du pointeur

filepos

```
// donne le numéro de l'article pointé par f
// ou le numéro du prochain article
function filepos(const f :<fichier>): CARDINAL;
```

Position du pointeur

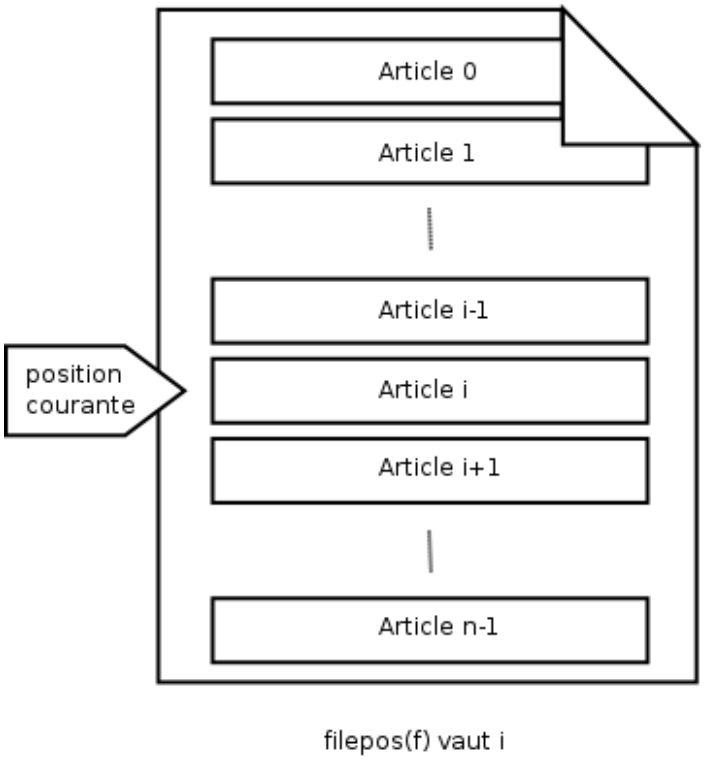

FIG.: La fonction **filepos**

Positionner le pointeur

seek

```
// place le pointeur sur l'article k
// CU : 0 ≤ k ≤ filesize(f)
procedure seek(const f : <fichier>;
               const k : CARDINAL);
```

Positionner le pointeur

seek

```
// place le pointeur sur l'article k
// CU : 0 ≤ k ≤ filesize(f)
procedure seek(const f : <fichier>;
               const k : CARDINAL);
```

Remarque : si $k = \text{filesize}(f)$, alors le pointeur est placé en fin de fichier.

Positionner le pointeur

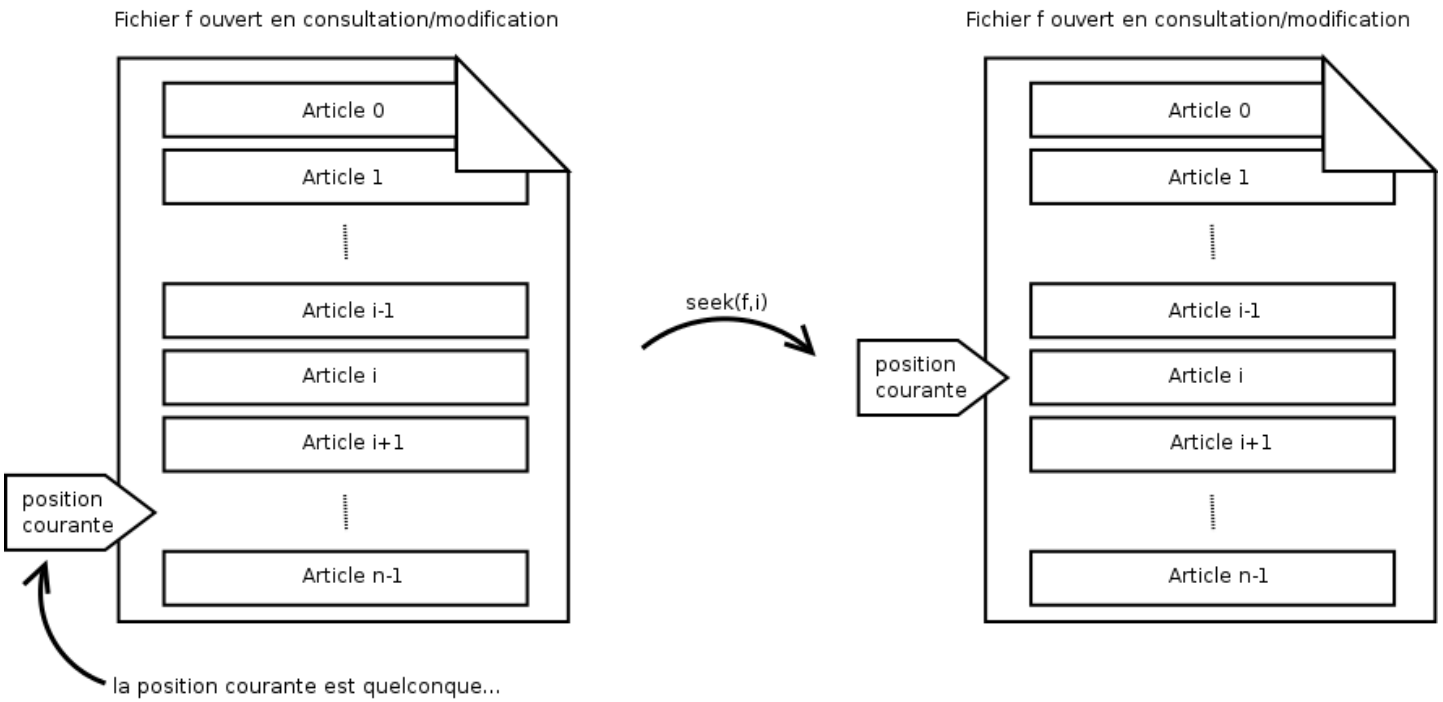

FIG.: La procédure **seek**

Ajout d'un article

Ajouter un article à la fin d'un fichier

```
6  program exemple3;
7
8  type
9     POINT = record
10       x,y : REAL;
11    end {POINT};
12
13    FICHIER_POINTS = file of POINT;
14
15 var
16    fp : FICHIER_POINTS;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    seek(fp,filesize(fp));
22    p.x := 2*pi;
23    p.y := 1.;
24    write(fp,p);
25    close(fp);
26 end.
```

Modification d'un article

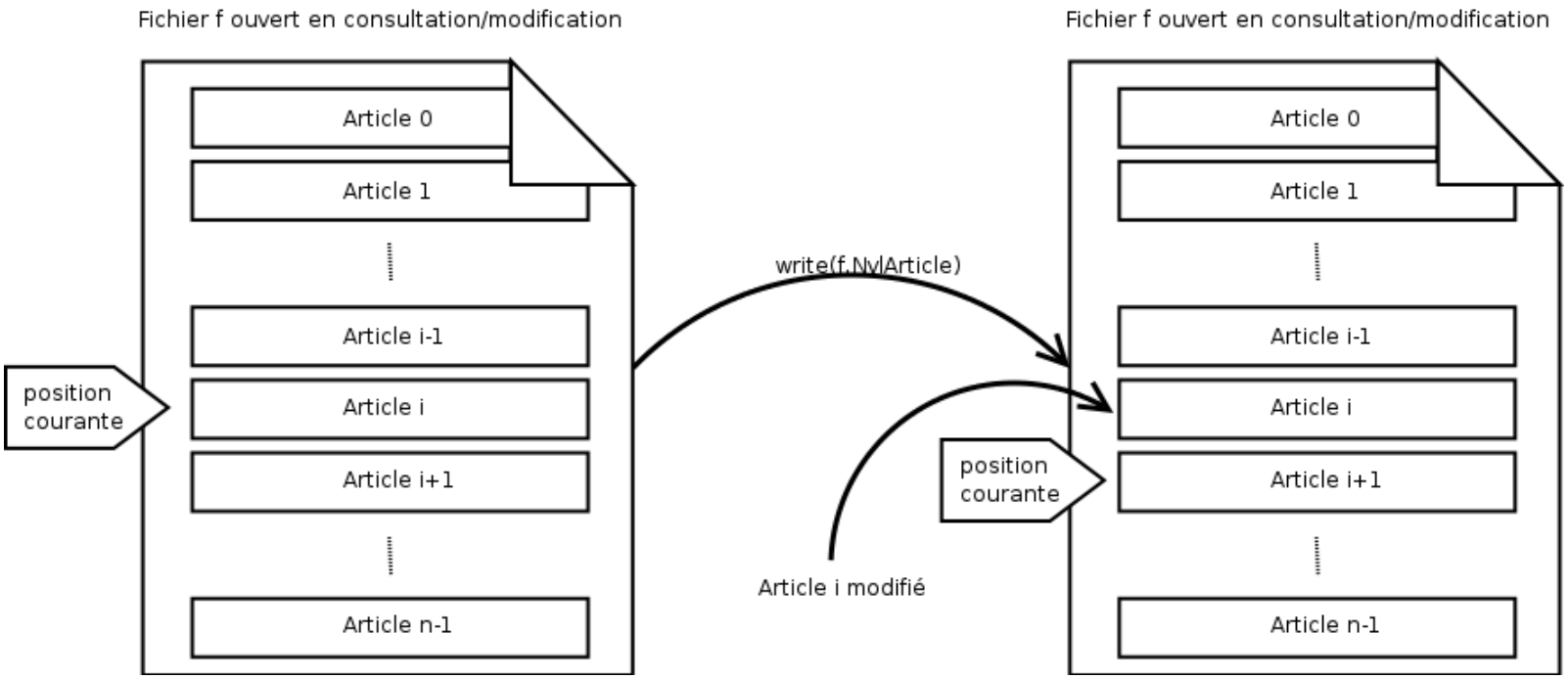

FIG.: Modification d'un article

Modification d'un article

Modifier le dernier article d'un fichier

```
6  program exemple4;
7
8  type
9     POINT = record
10       x,y : REAL;
11    end {POINT};
12
13    FICHIER_POINTS = file of POINT;
14
15 var
16    fp : FICHIER_POINTS;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    seek(fp,filesize(fp)-1);
22    p.x := 2*pi;
23    p.y := sin(p.x);
24    write(fp,p);
25    close(fp);
26 end.
```

Création d'un fichier

```
5  program exemple1;
6
7  type
8     POINT = record
9        x,y : REAL;
10    end {POINT};
11
12    FICHIER_POINTS = file of POINT;
13
14 var
15    fp : FICHIER_POINTS;
16    i : CARDINAL;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    rewrite(fp);
21    for i := 0 to 99 do begin
22       p.x := 2*i*pi/100;
23       p.y := sin(p.x);
24       write(fp,p);
25    end {for};
26    close(fp);
27 end.
```

Après exécution de ce programme, un fichier nommé `sinus.data` a été créé dans le répertoire courant. Ce fichier contient 100 articles (des points), chacun d'eux contenant deux REAL codés sur 8 octets. Le fichier a donc une taille de 1600 octets. Il n'est pas lisible avec un éditeur de texte car ce n'est pas un fichier texte.

Consultation d'un fichier

```
5  program exemple2;
6
7  type
8     POINT = record
9        x,y : REAL;
10    end {POINT};
11
12    FICHIER_POINTS = file of POINT;
13
14 var
15    fp : FICHIER_POINTS;
16    i : CARDINAL;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    i := 0;
22    while not eof(fp) do begin
23       read(fp,p);
24       writeln(i:3,p.x:7:3,p.y:7:3);
25       inc(i);
26    end {for};
27    close(fp);
28    writeln(i,' enregistrements lus');
29 end.
```

Ce programme ouvre le fichier `sinus.data` situé dans le répertoire courant, et affiche à l'écran son contenu.

Ajout d'un article

```
6  program exemple3;
7
8  type
9     POINT = record
10       x,y : REAL;
11    end {POINT};
12
13    FICHIER_POINTS = file of POINT;
14
15 var
16    fp : FICHIER_POINTS;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    seek(fp,filesize(fp));
22    p.x := 2*pi;
23    p.y := 1.;
24    write(fp,p);
25    close(fp);
26 end.
```

Après exécution de ce programme, le fichier nommé `sinus.data` situé dans le répertoire courant possède un article supplémentaire (incorrect). Ce fichier contient donc 101 articles.

Modification d'un article

```
6  program exemple4;
7
8  type
9     POINT = record
10       x,y : REAL;
11    end {POINT};
12
13    FICHIER_POINTS = file of POINT;
14
15 var
16    fp : FICHIER_POINTS;
17    p : POINT;
18 begin
19    assign(fp,'sinus.data');
20    reset(fp);
21    seek(fp,filesize(fp)-1);
22    p.x := 2*pi;
23    p.y := sin(p.x);
24    write(fp,p);
25    close(fp);
26 end.
```

Après exécution de ce programme, le dernier article du fichier nommé `sinus.data` a été modifié.