



# INF1005A: Programmation procedurale

## Chapitre 8: Lecture et écriture de fichiers



# Agenda

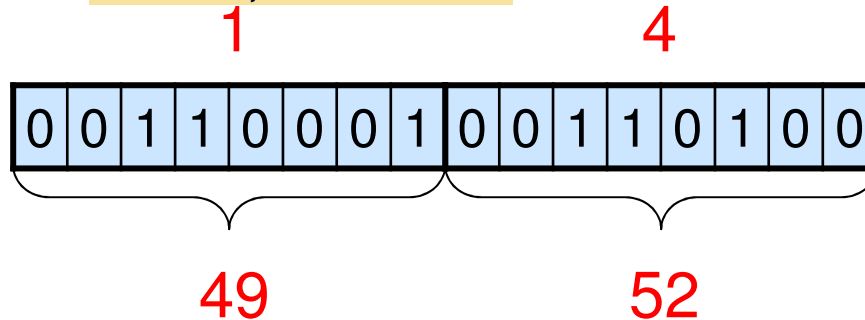
- Fichiers
- Fichiers texte
- Fichiers binaires
- Détection de la fin des fichiers
- Manipulation de fichiers et de répertoires
- Sommaire



# Fichiers – notions générales fichiers texte

## Fichiers texte

- fichiers séquentiels – accès séquentiel, sans accès direct aux données
- fin de ligne
- lus avec un éditeur de texte (notepad, wordpad...)
- codes ASCII de caractères uniquement
- i.e. le nombre 14 est écrit comme le code ASCII de 1 et le code ASCII de 4 : 49 52, en binaire



# Fichiers – notions générales fichiers binaires

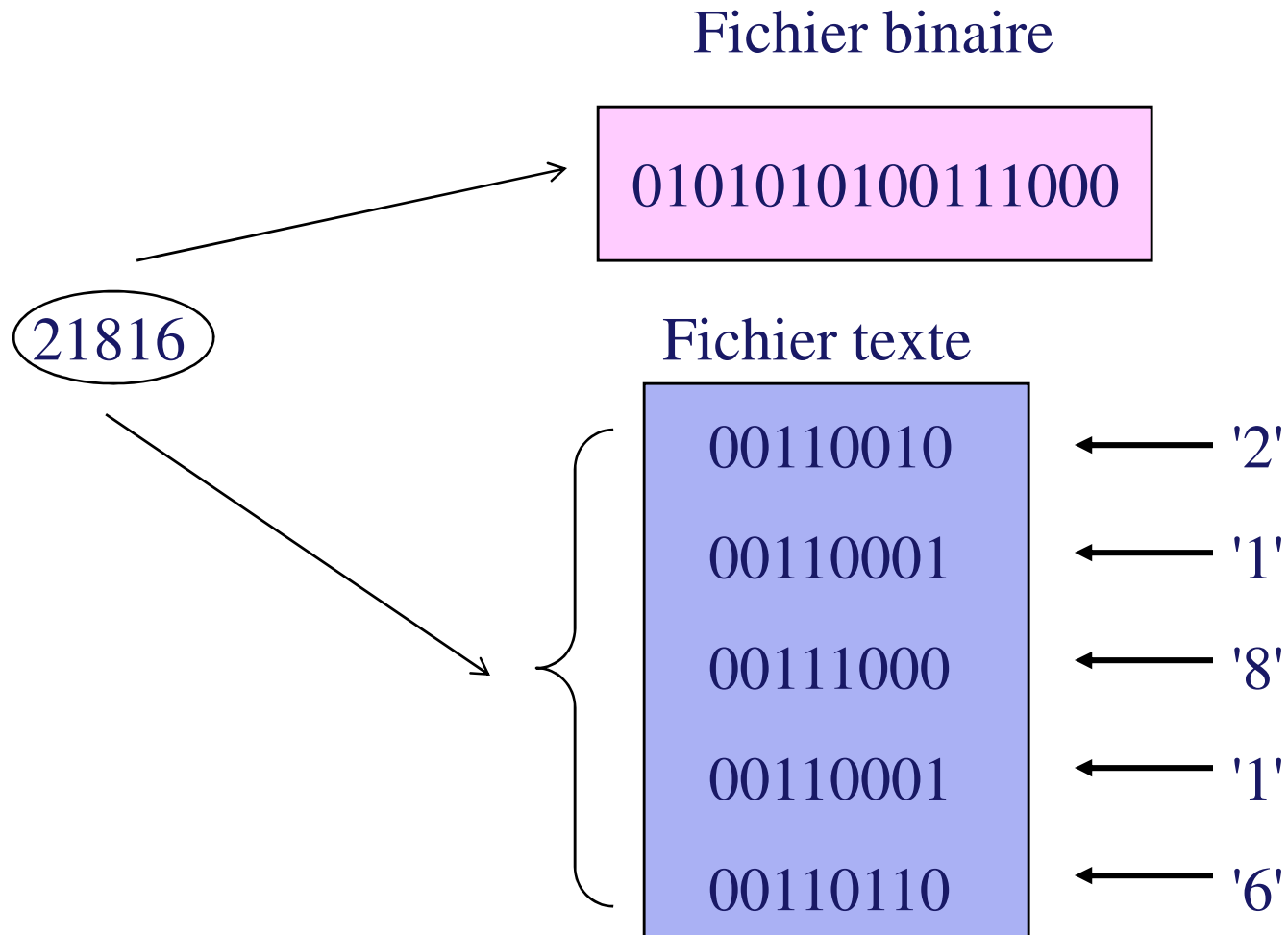
## Fichiers binaires

- accès aux données direct par un positionnement à l'emplacement désiré
- il n'y a pas de fin de ligne
- information sous forme codée, généralement inintelligible lorsqu'elle est affichée avec un éditeur de texte
- données dans les fichiers représentées de la même façon que les données en mémoire (Complément à 2, IEEE754)
  - i.e. le nombre 14 est écrit
- utilisent généralement moins d'espace

0	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---



# Fichiers – fichiers texte vs fichiers binaires



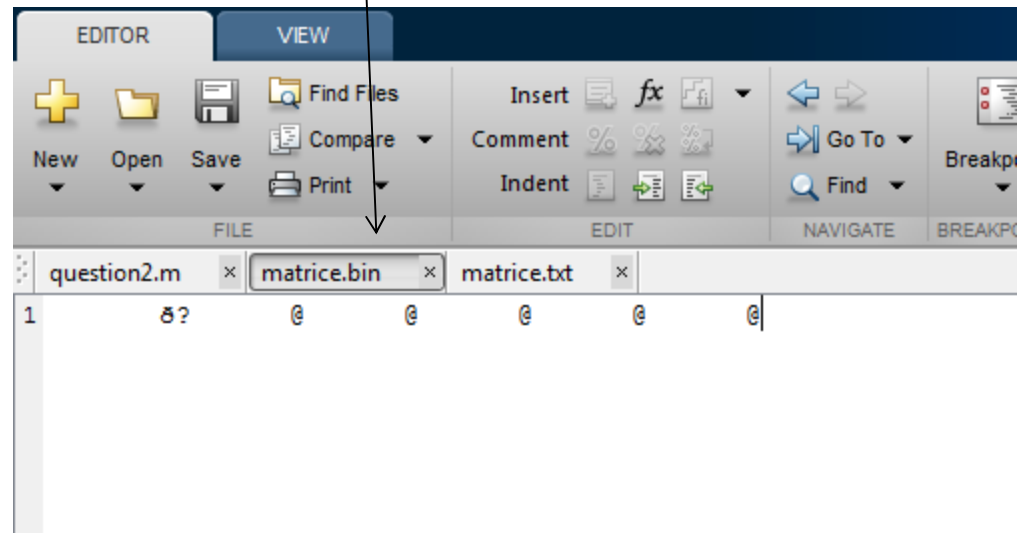
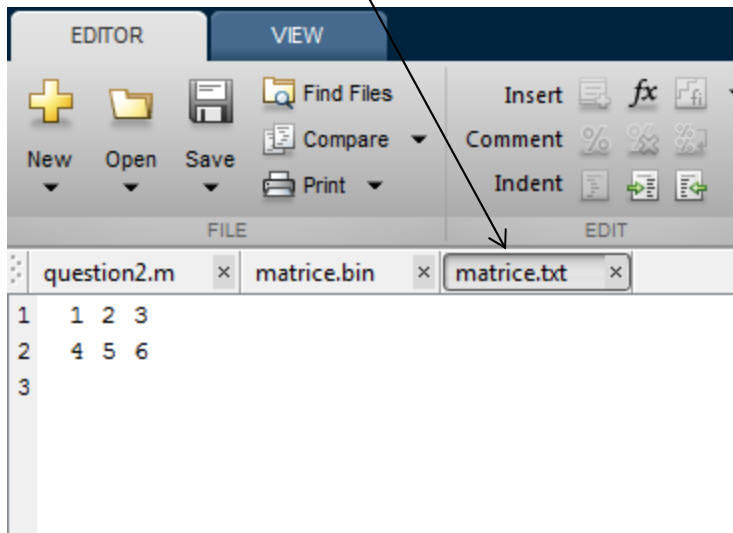
# Fichiers – fichiers texte vs fichiers binaires

La matrice [1 2 3;

4 5 6]

Fichier texte  
matrice.txt

Fichier binaire  
matrice.bin





# Fichiers – étapes manipulation fichiers texte et binaire

- Ouverture fichier(s)
- Vérification ouverture fichier(s)
- Traitement (lecture et/ou écriture fichier(s))
- Fermeture fichier(s)



### Fichiers – ouverture fichiers

**ID\_FIC** = **fopen**( 'NOM\_FIC' , 'TYPE\_OUVERTURE' )

**ID\_FIC** : identificateur du fichier;

- nombre entier, si égal à -1, l'ouverture a échoué.

**NOM\_FIC** : nom du fichier à ouvrir avec son extension;

- chaîne de caractères; il est aussi possible de spécifier le chemin (path) du fichier.

**TYPE\_OUVERTURE** : indique la façon dont le fichier sera ouvert (lecture ou écriture).

- chaîne de caractères, par défaut, le fichier sera ouvert en lecture.





### Fichiers – ouverture fichiers

Type	Description
r	Ouvre le fichier en lecture.
w	Ouvre le fichier en écriture. Si le fichier n'existe pas, il est créé. S'il existe, les données contenues sont effacées.
a	Ouvre le fichier en écriture. Si le fichier n'existe pas, il est créé. S'il existe, les données sont ajoutées à la suite des données déjà présentes dans le fichier.

- Ces types sont valables seulement pour les fichier binaires.
- Si le fichier est un fichier **texte**, ajouter **t** à chaque type.  
i.e. l'ouverture en lecture (r) deviendra **rt**.

### Fichiers – **vérification ouverture fichiers**

Après l'ouverture d'un fichier, il est important de toujours vérifier si l'ouverture s'est effectuée correctement.

Il faut vérifier que l'identificateur du fichier est différent de -1

Exemple:

```
fid=fopen('C:\documents\fic_test.txt','rt');  
if fid == -1  
    disp('Problème lors de l''ouverture');  
else  
    disp('Fichier ouvert');  
end
```



## Fichiers – fermeture des fichiers

Lorsqu'on a terminé de lire ou d'écrire dans un fichier, il faut toujours le fermer avec `fclose()` :

```
verification = fclose(ID_FIC)
```

`verification` : variable qui indique si la fermeture s'est effectuée avec problème ( -1 ) ou sans problème ( 0 ).

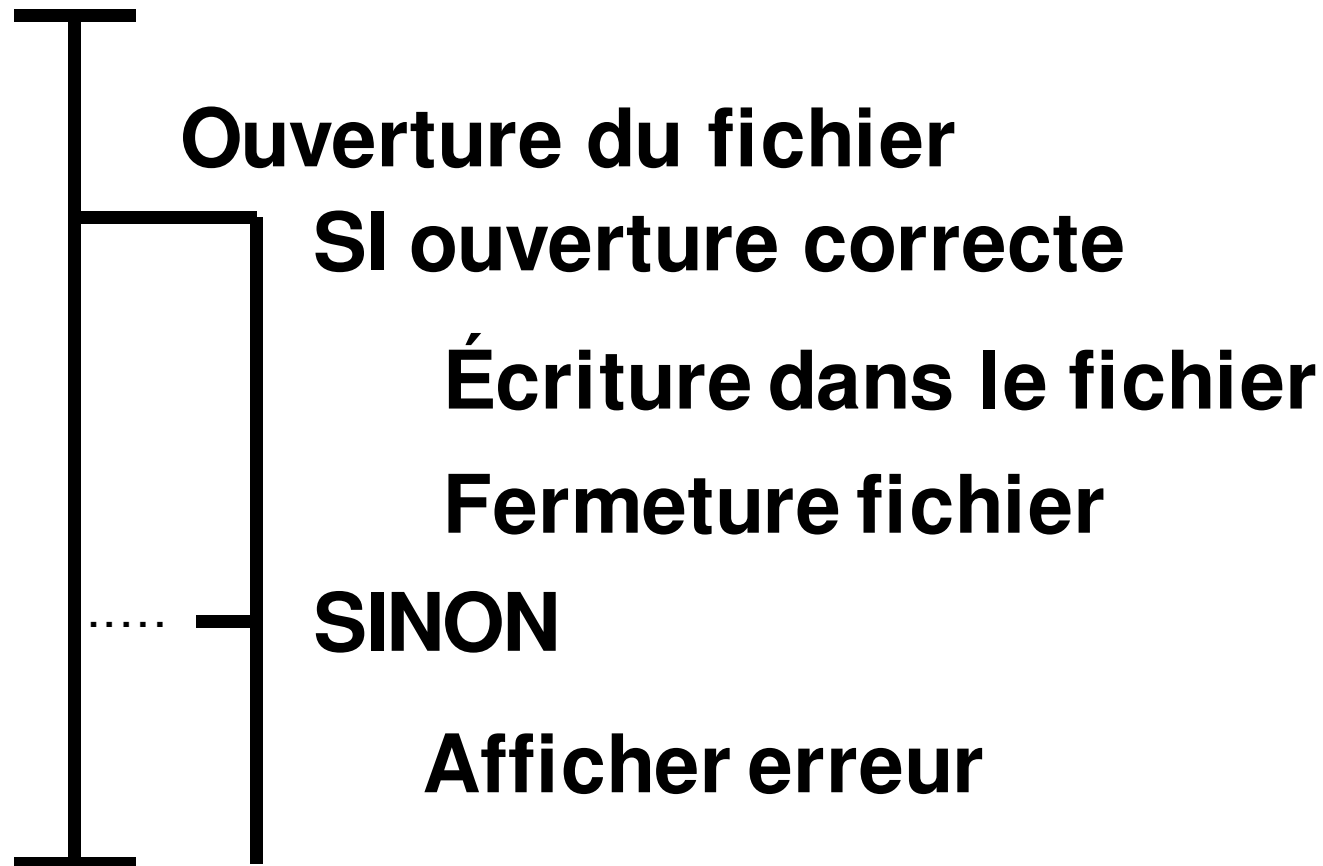
`ID_FIC` : identificateur du fichier ouvert.

Si plusieurs fichiers ont été ouverts, les fichiers doivent être fermés un à la fois.

**ATTENTION:** `fclose('all')` ferme tous les fichiers ouverts en MATLAB (dangereux donc l'utilisation de `fclose('all')` est Interdite).

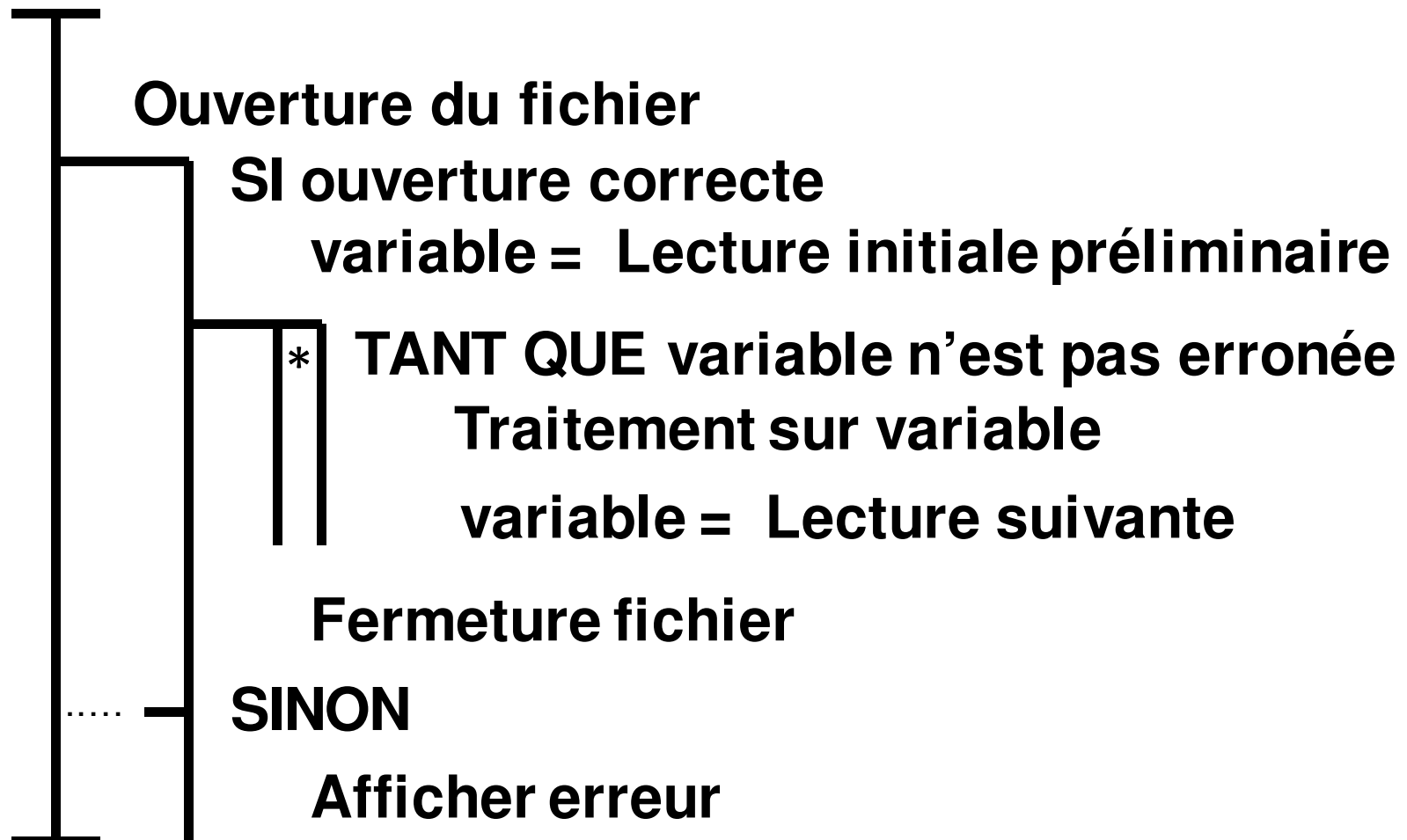


# Fichiers – pseudo-code général pour l'écriture dans un fichier





## Fichiers – pseudo-code général pour la lecture d'un fichier





# Agenda



## Fichiers

- Fichiers texte
- Fichiers binaires
- Détection de la fin des fichiers
- Manipulation de fichiers et de répertoires
- Sommaire

### Fichiers texte – écriture dans les fichiers texte (`fprintf()`)

Séquence d'opérations pour écrire dans un fichier texte:

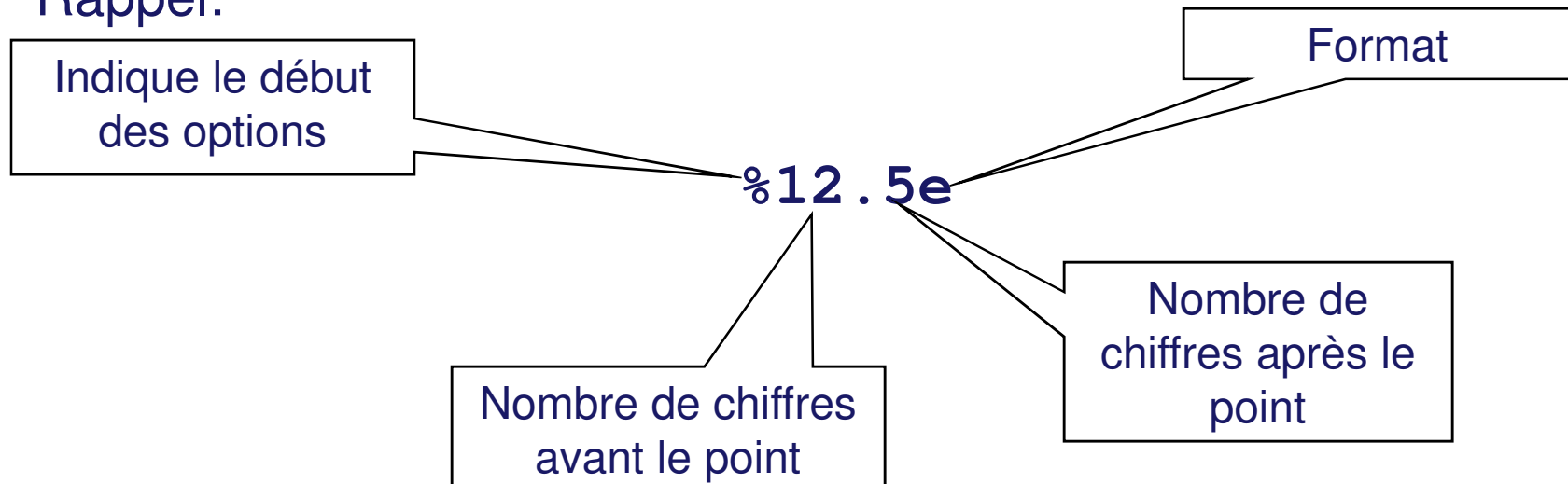
```
% ouverture fichier dont on écrit
Nofichier=fopen('NomdeFichier.txt','wt');
% vérification ouverture
if Nofichier ~= -1 % fichier ouvert
    ...
    % écriture dans un fichier texte
    fprintf(Nofichier,'Données à écrire avec %s, %d', var1,...,
    varN);
    ...
    fprintf(Nofichier,'Données à écrire avec %s, %d\n',var1,...,
    varN);
    ...
    % fermeture fichier
    fclose(Nofichier);
end
```

### Fichiers texte – écriture dans les fichiers texte (`fprintf()`)

```
var=fprintf(fid, 'Données+format', variable1, ...);
```

- fonction qui permet d'écrire dans un fichier texte.
- utilisée comme pour l'affichage à l'écran, sauf qu'on ajoute comme premier paramètre l'identificateur du fichier.

Rappel:





# Fichiers texte – écriture dans les fichiers texte (`fprintf()`)

Formats pouvant être utilisés avec `fprintf()` :

Format	Description
%c	Un caractère
%d	Notation d'un nombre décimal (signé)
%e	Notation exponentielle (utilisant un petit e dans 3.15e+00)
%E	Notation exponentielle (utilisant un grand E dans 3.15E+00)
%f	Notation point fixé
%g	Plus compact que %e et %f, n'affiche pas les zéros non significatifs
%G	Même que %g mais utilise un E majuscule
%i	Notation décimale d'un entier (signé)
%o	Notation octale
%s	Chaîne de caractères
%u	Notation décimale d'un entier (non signé)
%x	Notation hexadécimale (avec les lettres a-f en minuscule)
%X	Notation hexadécimale (avec les lettres A-F en majuscule)

# Fichiers texte – écriture dans les fichiers texte (`fprintf()`)

Caractères spéciaux utiles lors de l'écriture de fichiers texte.

Caractère	Description
<code>\b</code>	Efface le dernier caractère
<code>\f</code>	Changement de page
<code>\n</code>	Nouvelle ligne
<code>\r</code>	Retour de chariot
<code>\t</code>	Une tabulation horizontale
<code>\\</code>	Barre oblique
<code>\" ou \'</code>	Apostrophe
<code>%%</code>	Caractère pourcentage

```
A = 3.23;
```

```
fprintf(fid, 'Voici le contenu de A, %g \n', A);
```

# Fichiers texte – **lecture de fichiers texte – méthode 1** **(fscanf())**

Séquence d'opérations pour lire un fichier texte:

```
% ouverture fichier à lire
Nofichier=fopen('NomdeFichier.txt','rt');
% vérification ouverture
if Nofichier~= -1 % fichier ouvert
    ...
    % lecture d'un fichier texte
    variable = fscanf(Nofichier,'Données à lire avec %s, %d',
    [Nombre de données à lire]);
    ...
    variable = fscanf(Nofichier,'Données à lire avec %s, %d',
    [Nombre de données à lire]);
    ...
    % fermeture fichier
    fclose(NoFichier);
end
```



# Fichiers texte – **lecture de fichiers texte – méthode 1** **(fscanf())**

```
val=fscanf(ID_FIC, format, taille);
```

- fonction qui permet la lecture des éléments séparés par des espaces:

i.e. :

```
123 neige 4 % 3 éléments (mots) séparés par des  
% espaces
```

```
123 % le premier mot  
neige % le deuxième mot  
4 % le troisième mot
```



## Fichiers texte – **lecture de fichiers texte – méthode 1** **(fscanf())**

```
val=fscanf(ID_FIC, format, taille);
```

`val`: variable dans laquelle les données seront lues (sauvegardées).

`ID_FIC`: identificateur du fichier.

`format`: formats dans lesquels les différentes données lues seront transformées.

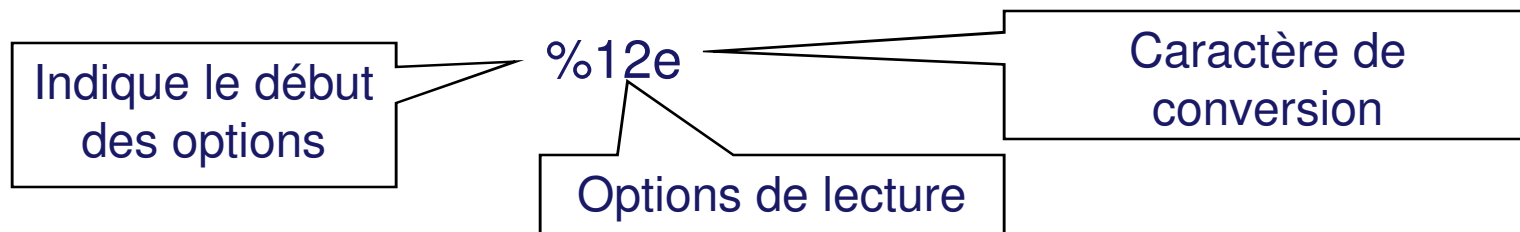
`taille`: indique le nombre d'éléments qui seront lus.

Différentes formes de `taille`:

<code>n</code>	Lecture de <code>n</code> éléments dans un vecteur colonne.
<code>inf</code>	Lecture jusqu'à la fin du fichier. Le résultat est un vecteur colonne avec le même nombre d'éléments que dans le fichier.
<code>[m,n]</code>	Lecture de suffisamment d'éléments pour remplir une matrice <code>m</code> par <code>n</code> . Lecture colonne par colonne. <code>n</code> peut être <code>inf</code> mais pas <code>m</code> .

# Fichiers texte – lecture de fichiers texte – méthode 1 (`fscanf()`)

Le format de lecture est spécifié comme suit:



On peut ajouter les caractères suivant entre `%` et le caractère de conversion :

*	Ignore la valeur correspondant au caractère de conversion.
Un nombre	La taille de la chaîne de caractères à lire.
Une lettre	Spécifie la taille du type lu comme <code>h</code> pour short ou <code>l</code> pour long.



# Fichiers texte – **lecture de fichiers texte – méthode 1** **(fscanf())**

Différents formats qui peuvent être utilisés avec `fscanf()` :

Caractère	Description
<code>%c</code>	Séquence de caractères, nombre spécifié entre % et c.
<code>%d</code>	Nombre décimal.
<code>%e, %f, %g</code>	Nombre réel.
<code>%i</code>	Entier signé en fonction de la base numérique utilisée dans le fichier.
<code>%o</code>	Entier octal signé.
<code>%s</code>	Une série de caractères autre que des espaces.
<code>%u</code>	Entier décimal signé.
<code>%x</code>	Entier hexadécimal signé.



### Fichiers texte – **lecture de fichiers texte – méthode 1** **(fscanf())**

**Remarque :** Il est possible de spécifier, en lecture, plusieurs formats mais ils doivent être du même type (tous numériques ou tous chaînes de caractères).

i.e. un fichier texte contenant: Taille 34.12352

```
Val = fscanf(fid, '%s %s', 2)  
Val contient 'Taille34.12352'
```

```
Val = fscanf(fid, '%s %f', 2)  
Val contient 84 97 105 108 108 101 34.12352
```

Les caractères sont convertis en nombre (leur code ASCII), car toutes les données d'une matrice (Val dans l'exemple) doivent être du même type.



# Fichiers texte – lecture de fichiers texte – méthode 2 (fget1())


Séquence d'opérations pour lire un fichier texte:

```
% ouverture fichier à lire
Nofichier=fopen('NomdeFichier.txt','rt');
% vérification ouverture
if Nofichier~= -1 % fichier ouvert
    ...
    % lecture d'un fichier texte
    variable = fget1(Nofichier);
    ...
    variable = fget1(Nofichier);
    ...
    % fermeture fichier
    fclose(Nofichier);
end
```



# Fichiers texte – **lecture de fichiers texte – méthode 2** **(fgetl())**

```
ligne=fgetl(ID_FIC);
```

- fonction qui permet la lecture d'une ligne complète (jusqu'à '`\n`') 
- la valeur de retour `ligne` est toujours une chaîne de caractères
- le fichier texte doit absolument contenir des '`\n`'

Soit le fichier texte contenant: Taille 34.12352

```
fid=fopen('Fichier.txt','rt');
```

```
Val=fgetl(fid); % Val contient 'Taille 34.12352'
```

```
fclose(fid);
```



# Agenda



Fichiers



Fichiers texte

- Fichiers binaires
- Détection de la fin des fichiers
- Manipulation de fichiers et de répertoires
- Sommaire

# Fichiers binaires – écriture dans les fichiers binaires (**fwrite()**)

Séquence d'opérations pour écrire dans un fichier binaire:

```
% ouverture fichier dont on écrit
Nofichier=fopen('NomdeFichier.bin','w');
% vérification ouverture
if Nofichier ~= -1 % fichier ouvert
    ...
    % écriture dans un fichier binaire
    fwrite(Nofichier,[variable], 'type variable');
    ...
    fwrite(Nofichier,[variable], 'type variable');
    ...
    % fermeture fichier
    fclose(NoFichier);
end
```

# Fichiers binaires – écriture dans les fichiers binaires (`fwrite()`)

```
var = fwrite(ID_FIC, données, 'type');
```

➤ fonction qui permet d'écrire dans un fichier binaire.

`compteur` : nombre d'éléments écrits dans le fichier. S'il n'est pas identique au nombre d'éléments à écrire, un problème est survenu lors de l'écriture du fichier.

`ID_FIC` : identificateur du fichier.

`données` : variable contenant les données à écrire dans le fichier

`type` : type dans lequel les données seront écrites (*voir la page suivante*).

## Fichiers binaires – écriture dans les fichiers binaires (`fwrite()`)

```
var=fwrite(ID_FIC, données, 'type');
```

➤ les différents types possibles avec MATLAB:

Type	Description
char	Caractère (8 bits)
int8	Entier (8 bits)
int16	Entier (16 bits)
int32	Entier (32 bits)
int64	Entier (64 bits)
uint8	Entier non-signé (8 bits)
uint16	Entier non-signé (16 bits)
uint32	Entier non-signé (32 bits)
uint64	Entier non-signé (64 bits)
float32	Nombre réel (32 bits)
float64	Nombre réel (64 bits)
double	Nombre réel (64 bits)

# Fichiers binaires – écriture dans les fichiers binaires (`fwrite()`)

```
var=fwrite(ID_FIC, données, 'type');
```

➤ exemple: les différents types possibles avec MATLAB:

i.e. une matrice à écrire: `matrice = [1 2 4 8 1 3];`

```
fid=fopen('NomdeFichier.bin','w');  
compteur = fwrite(fid,matrice,'double'); % compteur=6  
fclose(fid)
```

**NOTE (très importante):**  
Les données de `matrice` sont  
écrites colonne par colonne

### Fichiers binaires – lecture des fichiers binaires (`fread()`)

Séquence d'opérations pour lire un fichier binaire:

```
% ouverture fichier à lire
Nofichier=fopen('NomdeFichier.bin','r');
% vérification ouverture
if Nofichier~= -1 % fichier ouvert
    ...
    % lecture d'un fichier texte
    variable = fread(Nofichier,[nombre de données], 'type');
    ...
    variable = fread(Nofichier,[nombre de données], 'type');
    ...
    % fermeture fichier
    fclose(NoFichier);
end
```



### Fichiers binaires – **lecture des fichiers binaires (fread())**

```
var=fread(ID_FIC, taille, 'type');
```

- fonction qui permet la lecture d'un fichier binaire.

`var` : variable dans laquelle les données seront lues (sauvegardées).

`ID_FIC` : identificateur du fichier.

`taille` : nombre d'éléments qui seront lus. Si absent, le fichier sera lu complètement. Il est possible de spécifier la taille de la matrice de sortie. Si la taille est supérieure au nombre de données lues, des 0 seront insérés dans les cases vides.

`type` : type des données lues (*voir les types donnés dans le tableau de types de `fwrite()`*).



# Fichiers binaires – lecture des fichiers binaires (`fread()`)

Exemple:

```
matrice = fread(fid, [3,3], 'double')
```


```
matrice =  1  3  6  
          2  7  0  
          1  7  0
```

Remplissage dû à un nombre  
insuffisant de données (le fichier  
contenait 7 nombres).

Taille de la matrice dans  
laquelle on lit (sauvegarde)  
les données.


**NOTE (très importante):**  
Les données sont lues  
colonne par colonne.

# Fichiers binaires – lecture des fichiers binaires (`fread()`)

IMPORTANT: Pour lire un ou des caractères dans un fichier binaire, il faut convertir leurs codes ASCII en caractères, car `fread()` ne donne en sortie que des matrices de nombres. 

Exemple pour un fichier binaire contenant le caractère 'a' :

```
>> matrice = fread(fid,1,'char')  
matrice = 97      %(le code ASCII de 'a')
```

```
>> matrice = fread(fid,1,'char=>char')   
matrice = 'a'
```

```
>> matrice = fread(fid,1,'*char')  
matrice = 'a'
```

La conversion est faite avec `=>type` ou `*type` où `type` est l'un de ceux de la page 30. (i.e. 'char=>char')



## Fichiers binaires – déplacement dans les fichiers binaires (`fseek()`)

```
var=fseek(ID_FIC, déplacement, 'origine');
```

- fonction qui permet le déplacement dans un fichier binaire.

`var` : permet de vérifier si le déplacement a été effectué avec erreur (`var = -1`) ou sans erreur (`var = 0`).

`ID_FIC` : identificateur du fichier.

`déplacement` : nombre d'octets de déplacement.

Si positif, le déplacement se fera vers la fin du fichier,  
si négatif, il se fera vers le début du fichier.

`origine` : indique à partir de quelle position le déplacement sera effectué.



## Fichiers binaires – déplacement dans les fichiers binaires (`fseek()`)

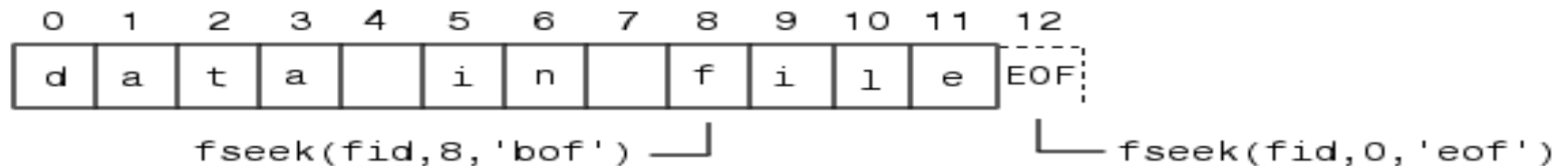
```
var=fseek(ID_FIC, déplacement, 'origine');
```

`origine` : indique à partir de quelle position, le déplacement sera effectué.

Abréviation	Description
bof	Début du fichier.
cof	Position courante du curseur dans le fichier.
eof	Fin du fichier.

```
verification = fseek(fid,8,'bof')
```

```
verification = 0
```





## Fichiers binaires – déplacement dans les fichiers binaires (`ftell()`)

```
position=ftell(ID_FIC);
```

- fonction qui permet de connaître le nombre d'octets entre la position courante du curseur de lecture de fichier et le début du fichier.

`position` : variable contenant le nombre d'octets entre le début du fichier et la position courante du curseur.

`ID_FIC` : identificateur du fichier.



## Fichiers binaires – déplacement dans les fichiers binaires (`fseek()` et `ftell()`)

En combinant les fonctions `fseek()` et `ftell()`, il est possible d'effectuer plusieurs opérations.

Une opération possible est de déduire la taille d'un fichier:

```
verif = fseek(fid,0,'eof');  
if verif == 0  
    taille_fic = ftell(fid);  
else  
    error('Probleme lors du deplacement');  
end
```

Contient la  
taille en octets  
du fichier



# Agenda



Fichiers



Fichiers texte



Fichiers binaires

- Détection de la fin des fichiers
- Manipulation de fichiers et de répertoires
- Sommaire



### Détection de la fin des fichiers

Pour savoir si on a lu un fichier au complet ou non, on doit vérifier si la fonction de lecture retourne une valeur erronée:

`fgetc()` donne `-1`, lorsqu'il y a une erreur de lecture (utilisation de `ischar()`);

`fscanf()` donne `[ ]` ou `' '` selon le format de lecture, lorsqu'il y a une erreur de lecture (utilisation de `isempty()`);

`fread()` donne `[ ]`, lorsqu'il y a une erreur de lecture (utilisation de `isempty()`);

Une erreur de lecture inclut l'atteinte de la fin du fichier. Donc, il faut utiliser les valeurs de retour de ces fonctions pour détecter la fin d'un fichier dans un **while**.

# Détection de la fin des fichiers

Une alternative non-recommandée est de vérifier l'indicateur de fin de fichier.

```
FinFichier = feof(ID_FIC)
```

`FinFichier` : variable qui indique si la fin du fichier a été atteinte (1) ou non (0).

`ID_FIC` : identificateur du fichier ouvert.

`feof()` indique que la fin d'un fichier a été atteinte seulement si on tente de lire après la fin du fichier.

Tout comme la première méthode, `feof()` *doit être jumelé* à une boucle **while**.

À éviter pour deux raisons principales:

- Nécessite une lecture préliminaire pour bien fonctionner et aucune erreur n'est émise dans le cas échéant;
- Problème avec `fgetl()` *lors de la lecture de la dernière phrase*



# Agenda



Fichiers



Fichiers texte



Fichiers binaires



Détection de la fin des fichiers

- Manipulation de fichiers et de répertoires
- Sommaire



# Manipulations de fichiers et de répertoires - **fonctions**

`movefile()` changer le nom d'un fichier ou d'un répertoire.  
`movefile('nom_précédent', 'nouveau_nom')`

`mkdir()` créer un nouveau répertoire.  
`mkdir('nom_répertoire')`

`delete()` détruire un fichier.  
`delete('nom_fichier')`

`rmdir()` détruire un répertoire.  
`rmdir('nom_répertoire')`



# Agenda



Fichiers



Fichiers texte



Fichiers binaires



Détection de la fin des fichiers



Manipulation de fichiers et de répertoires

- Sommaire

### Sommaire

- Avant toutes opérations sur un fichier texte ou binaire, il faut ouvrir le fichier avec `fopen()`.
- Les données du fichier texte sont lues avec `fscanf()` ou `fgetl()`.
- Les informations sont écrites dans un fichier texte avec `fprintf()`.
- Pour les fichiers binaires, `fread()` est utilisée pour lire les données.
- Pour écrire les fichiers binaires, la fonction `fwrite()` est utilisée.
- Il est parfois utile de se déplacer dans le fichier binaire avec `fseek()`.
- Fonction `ftell()` peut donner la dimension d'un fichier binaire.
- Après avoir effectué toutes les opérations souhaitées, le fichier doit être fermé obligatoirement avec `fclose()`.



# Sommaire – comparaison entre les fichiers texte et les fichiers binaires

	fichier texte	fichier binaire
extension	.txt	.bin .dat
ouverture	fopen(nom_fichier.extension, type_ouverture)	
types ouverture	rt', 'wt', 'at'	r', 'w', 'a'
validation ouverture	identificateur ~=-1	
écriture	fprintf()	fwritef()
lecture	fscanf(), fgetl()	fread()
fermeture fichier	fclose()	
type de données	ASCII seulement	représentées de la même façon que les données en mémoire (Complément à 2, IEEE754)
propriétés	il y a fin de ligne	fin de ligne n'existe pas
	peut être lu avec un éditeur de texte	généralement inintelligible lorsqu'on affiche avec un éditeur de texte
	fichier séquentiel, sans accès direct aux données	l'accès aux données peut être direct par un positionnement à l'emplacement désiré (la fonction fseek())



# Agenda



Fichiers



Fichiers texte



Fichiers binaires



Détection de la fin des fichiers



Manipulation de fichiers et de répertoires



Sommaire





# Sommaire

- 1 Fichiers
- 2 Fichiers texte
- 3 Fichiers binaires
- 4 Détection de la fin du fichier
- 5 Manipulation de fichiers et de répertoires
- 6 Sommaire