



# Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fichiers](#) ► [Opérations sur les fichiers](#)

## Opérations sur les fichiers

Pour réaliser certaines opérations de lecture/écriture sur un fichier nous allons avoir besoin d'un "descripteur". Ce descripteur est en fait un repère interne à notre fichier. Pour obtenir un descripteur de fichier nous allons utiliser la fonction [fopen\(\)](#). Cette fonction renvoie un "descripteur de fichier" en "ouvrant" un fichier. L'ouverture de ce fichier peut se faire selon plusieurs "mode" (tableau extrait de la documentation) :

Mode	Description
'r'	Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
'r+'	Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
'w'	Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'w+'	Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'a'	Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'a+'	Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'x'	Crée et ouvre le fichier en écriture seulement ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, fopen() va échouer, en retournant FALSE et en générant une erreur de niveau E_WARNING. Si le fichier n'existe pas, fopen() tente de le créer.
'x+'	Crée et ouvre le fichier pour lecture et écriture; le comportement est le même que pour 'x'.

Le descripteur est en fait un repère qui se positionne à l'endroit où va avoir lieu la lecture/écriture. Par exemple nous allons ouvrir le fichier "haiku.txt" en lecture seule :

```
1 <?php
2
3 // operations_fichiers_1.php
4 $fd = fopen('haiku.txt', 'r');
```

Le descripteur est placé à la position 0 de notre fichier c'est-à-dire devant le premier octet :

```
Je lève la tête
L'arbre que j'abats
Comme il est calme
Issekiro
```

Le curseur représente le descripteur. La fonction [fclose\(\)](#) permet de fermer le fichier et de "libérer" la ressource.

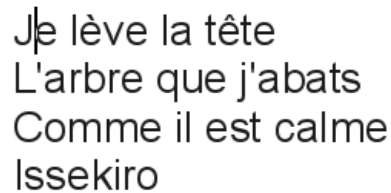
La fonction [fgetc\(\)](#) permet de lire un caractère à partir d'un descripteur. À chaque appel de [fgetc\(\)](#), le descripteur avancera d'un cran (un octet) et renverra le caractère lu :

```

1 <?php
2
3 // operations_fichiers_2.php
4 $fd = fopen('haiku.txt', 'r');
5 $car = fgetc($fd);
6 echo $car;
7 fclose($fd);

```

L'exécution déplacera le descripteur et affichera "J" :



Je lève la tête  
L'arbre que j'abats  
Comme il est calme  
Issekiro

La fonction `fgetc()` renvoie `false` quand il n'y a plus rien à lire. `fgetc()` renvoie donc soit un caractère, soit `false`. Si je souhaite lire tous les caractères de mon fichier je dois donc afficher le caractère récupéré tant que la valeur que me renvoie `fgetc()` est différent de `false`. Je pourrais donc utiliser une boucle `while` avec pour condition que le caractère que me renvoie `fgetc()` soit différent de `false`. Je serais tenté d'écrire cela :

```

1 <?php
2
3 // operations_fichiers_3.php
4 $fd = fopen('haiku.txt', 'r');
5 while (fgetc($fd) !== false) {
6     // afficher caractère
7 }
8 fclose($fd);

```

Si `fgetc()` renvoie un caractère, j'entrerai dans ma boucle et je devrai afficher ce caractère. Seulement je n'ai pas stocké le caractère lu par `fgetc()`, et si je rappelle `fgetc()` j'obtiendrai le caractère suivant. Il me faudrait donc stocker dans une variable la valeur retournée par `fgetc()` dans la condition du `while`. Je vais donc, dans la condition du `while`, appeler `fgetc()`, stocker la valeur de retour de `fgetc()` dans une variable. Ce qui donne :

```

1 <?php
2
3 // operations_fichiers_4.php
4 $fd = fopen('haiku.txt', 'r');
5 while (($car = fgetc($fd)) !== false) {
6     echo $car;
7 }
8 fclose($fd);

```

Tout d'abord j'ouvre mon fichier en lecture. Ensuite, dans la condition du `while`, j'assigne à `car` la valeur de retour de `fgetc()`. Si `fgetc()` a réussi à lire un caractère, alors ce caractère a été assigné à `car`, donc `car` a une valeur qui n'est pas de type booléen : la condition est donc vraie et nous entrons dans la boucle. Par contre si `fgetc()` est arrivé à la fin du fichier, il a renvoyé `false`, la valeur assignée à `car` est donc `false` et la boucle s'arrête.

Nous pourrions faire la même chose en utilisant cette fois la fonction `fgets()` qui lit une ligne entière et qui renvoie `false` à la fin de la lecture.

```

1 <?php
2
3 // operations_fichiers_5.php
4 $fd = fopen('haiku.txt', 'r');
5 while (($line = fgets($fd)) !== false) {
6     echo $line;
7 }
8 fclose($fd);

```

D'autres fonctions de lecture sont utilisables :

- [fread\(\)](#) : permet de lire un certain nombre d'octet passé en argument à partir d'un descripteur
- [stream\\_get\\_contents\(\)](#) renvoie tout le contenu dans une chaîne de caractères, équivalent à [file\\_get\\_contents\(\)](#) mais travaille à partir d'un descripteur
- [fpassthru\(\)](#) : affiche le contenu à partir d'un descripteur
- [fgetcsv\(\)](#) : permet de lire une ligne d'un fichier csv, renvoie un tableau contenant les valeurs

Pour écrire dans un fichier il suffit d'obtenir un descripteur avec un mode qui permet l'écriture, soit en écrasant la donnée présente, soit en ajoutant à la fin du fichier. La fonction [fwrite\(\)](#) (ou [fputs\(\)](#) qui est un alias) écrit une chaîne de caractères à partir d'un descripteur. Si je souhaite ajouter un autre haïku après celui d'Issekiro, je peux écrire cela :

```

1 <?php
2
3 // operations_fichiers_6.php
4 $data = <<<END
5 Ce chemin
6 personne ne le prend
7 que le couchant d'automne
8 Basho
9 END;
10 $fd = fopen('haiku.txt', 'ab');
11 fwrite($fd, $data);
12 fflush($fd);
13 fclose($fd);

```

Dans le mode d'ouverture du fichier nous avons ajouté en dernier la lettre 'b' qui signifie l'ouverture du flux en mode binaire (recommandé par PHP). De plus, après l'écriture nous avons appelé la fonction [fflush\(\)](#) qui permet de vider le buffer d'écriture.

D'autres fonctions permettent de récupérer des informations sur le fichier, notamment :

- [dirname\(\)](#) qui renvoie le nom du dossier parent
- [realpath\(\)](#) qui renvoie le chemin absolu

La fonction [parse\\_ini\\_file\(\)](#) permet d'analyser un [fichier de configuration](#) .ini et renvoie le résultat dans un tableau. La fonction [touch\(\)](#) permet de créer ou de mettre à jour les dates de modification et d'accès. Enfin, la fonction [unlink\(\)](#) permet de supprimer un fichier.

Fin

## NAVIGATION

### Accueil

#### ■ Ma page

Pages du site

Mon profil

Cours actuel

#### Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle


Les structures de données


Les fonctions

Les erreurs

Les fichiers

 [Inclusion de fichiers](#)

 [Lecture rapide de fichier](#)

 [Écriture rapide de fichier](#)

 [Opérations sur les fichiers](#)

[Opérations sur les répertoires](#)[Les constantes magiques](#)[T.P. fonction copier\(\)](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

## ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))  
[Intro PHP](#)