

# **La manipulation des fichiers en PHP**

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Les fonctions de base de la manipulation des fichiers en PHP</b>	<b>3</b>
A. Les fonctions de base de la manipulation des fichiers en PHP .....	3
B. Exercice : Quiz .....	7
<b>III. La manipulation de fichiers en PHP : copy(), rename(), unlink() et file_exists()</b>	<b>8</b>
A. La manipulation de fichiers en PHP : copy(), rename(), unlink() et file_exists().....	8
B. Exercice : Quiz .....	12
<b>IV. Essentiel</b>	<b>13</b>
<b>V. Auto-évaluation</b>	<b>13</b>
A. Exercice .....	13
B. Test .....	14
<b>Solutions des exercices</b>	<b>15</b>

## I. Contexte

- **Durée** : 1 heure
- **Prérequis** : bases de PHP
- **Environnement de travail** : IDE - WampServer - PHP 8.1.xx - phpmyadmin5.2.1

### Contexte

PHP est un langage de programmation très populaire pour la création de sites web dynamiques. Au cours de ce processus, la manipulation de fichiers est une tâche essentielle. La capacité de stocker et de récupérer des données sur le disque dur d'un serveur est importante pour la gestion de sites internet et pour assurer leur bon fonctionnement. Pour faciliter la manipulation des fichiers, PHP offre des fonctions puissantes pour gérer toutes ces opérations. Les fonctions de manipulation de fichiers en PHP comprennent la lecture et l'écriture de données dans un fichier, ou la modification / suppression de fichiers existants.

Dans la première partie de notre cours, nous allons nous concentrer sur ces fonctions de base, nous allons explorer comment créer, ouvrir et fermer des fichiers, et également examiner les fonctions pour renommer, copier, supprimer et vérifier l'existence de fichiers. Enfin, nous allons aussi voir comment travailler avec les répertoires pour créer, renommer, copier et supprimer des dossiers.

## II. Les fonctions de base de la manipulation des fichiers en PHP

### A. Les fonctions de base de la manipulation des fichiers en PHP

En PHP, il existe différents modes d'ouverture de fichier qui permettent de spécifier la manière dont le fichier sera ouvert et utilisé, les plus courants sont les suivants.

#### Le mode d'ouverture de fichiers

Le mode d'ouverture de fichiers est un concept clé dans la manipulation de fichiers pour n'importe quel langage de programmation, y compris en PHP. Ce mode est spécifié dans le deuxième argument de la fonction `fopen()`, qui indique comment le fichier doit être ouvert pour être lu ou écrit. Il existe différents modes d'ouverture de fichiers, chacun étant utilisé en fonction de l'objectif de la manipulation de fichiers. Les modes d'ouverture les plus couramment utilisés sont :

- **"r" (read)** : ce mode permet de lire un fichier existant. Il est utilisé lorsque l'on souhaite simplement lire le contenu du fichier sans le modifier. Si le fichier n'existe pas, la fonction `fopen()` retourne « false ».
- **"w" (write)** : ce mode permet d'écrire dans un fichier. Il est utilisé pour créer un nouveau fichier ou écraser le contenu d'un fichier existant. Si le fichier n'existe pas, la fonction `fopen()` tente de le créer. Si la création échoue, la fonction retourne « false ».
- **"a" (append)** : ce mode permet d'ajouter du contenu à un fichier existant. Le contenu est ajouté à la fin du fichier sans effacer son contenu existant. Si le fichier n'existe pas, la fonction `fopen()` tente de le créer. Si la création échoue, la fonction retourne « false ».

Il est important de choisir le mode d'ouverture approprié pour chaque opération de lecture ou d'écriture de fichiers. En outre, il est essentiel de prendre en compte les permissions d'accès du fichier lors de l'ouverture, notamment pour les opérations de lecture et d'écriture simultanées.

#### La lecture de fichier - `fread()`

Tout d'abord, nous allons voir la fonction de lecture `fread()`. Cette fonction prend 2 paramètres en entrée : la ressource de fichier retournée par la fonction `fopen()`, et la taille en octets du contenu à lire. L'utilisation de la fonction `fread()` permet de lire des fichiers de manière efficace, car elle permet la lecture par blocs de taille

spécifiée. Une fois les données lues, elle les retourne sous forme de chaîne de caractères, ce qui les rend faciles à manipuler pour des traitements ultérieurs. De ce fait, cela est très utile pour la manipulation de fichiers, tels que la lecture de fichiers journaux, de fichiers texte voire CSV, ou encore la lecture de fichiers binaires. C'est une fonction importante et utilisée dans de nombreux types d'applications, il est donc important de la comprendre et de l'utiliser efficacement.

#### Exemple

```
1 $file = fopen("monfichier.txt", "r");
2 $content = fread($file, filesize("monfichier.txt"));
3 fclose($file);
```

On observe dans cet exemple l'utilisation de plusieurs fonctions pour la manipulation d'un fichier en PHP. Premièrement, la fonction `fopen()` est appelée avec le nom du fichier (qui peut également être une URL dans d'autres cas) et le mode de lecture spécifié ("r"). Ensuite, la fonction `fread()` est utilisée pour lire tout le contenu du fichier ouvert en se basant sur sa taille (obtenue grâce à la fonction `filesize()`). Enfin, la fonction `fclose()` est appelée pour fermer le fichier ouvert et libérer les ressources utilisées. Voilà un exemple simple, mais utile de l'utilisation de ces fonctions. Par ailleurs, si l'on décide d'utiliser la fonction `echo` sur la variable `$content`, le contenu de "monfichier.txt" s'exécutera côté HTML :

```
1 $file = fopen("monfichier.txt", "r");
2 $content = fread($file, filesize("monfichier.txt"));
3 fclose($file);
4 echo $content;
```

#### La lecture de fichier - `fgets()`

Dans les fonctions permettant la lecture de fichiers, on retrouve la fonction `fgets()` qui peut être très utile. Elle permet de lire une ligne à partir d'un fichier en utilisant une ressource de fichier retournée par la fonction `fopen()`. La fonction renvoie une chaîne de caractères correspondant à la ligne lue jusqu'au caractère de fin de ligne ou jusqu'à la fin du fichier. Un exemple concret d'utilisation de la fonction `fgets()` pourrait être de lire un fichier texte contenant des noms et des prénoms, ligne par ligne, pour les afficher ensuite dans une page web. On pourrait utiliser la fonction `fgets()` pour lire chaque ligne du fichier, stocker les informations dans des variables et les afficher à l'écran.

Voici un exemple d'utilisation de la fonction `fgets()` :

```
1 $file = fopen("monfichier.txt", "r");
2 if ($file) {
3     while (($line = fgets($file)) !== false) {
4         echo $line;
5     }
6     fclose($file);
7 }
```

#### Exemple

Dans l'exemple présenté, le fichier "monfichier.txt" est en mode lecture ("r"), la fonction `fgets()` est utilisée à l'intérieur d'une boucle `while` pour lire chaque ligne du fichier. La boucle se poursuit jusqu'à la fin du fichier, détectée par la valeur « false » renvoyée par la fonction `fgets()`. Enfin, la fonction `fclose()` est appelée pour fermer le fichier. C'est une fonction très pratique pour lire des fichiers volumineux, car elle permet de ne charger en mémoire qu'une seule ligne à la fois, ce qui peut réduire considérablement l'espace alloué nécessaire pour traiter le fichier.

Une autre fonction de lecture est la fonction `file_get_contents()`. Elle permet de lire le contenu d'un fichier et de le renvoyer sous forme de chaîne de caractères. Elle prend un seul paramètre, qui est le chemin du fichier à lire. C'est pratique pour lire des petits fichiers, comme des fichiers de configuration ou des fichiers de texte. Voici un exemple d'utilisation de la fonction :

```
1 $contents = file_get_contents('monfichier.txt');
2 echo $contents;
```

### L'écriture - `fwrite()`

Maintenant que nous avons vu les fonctions de lecture, intéressons-nous aux fonctions d'écriture. Ces fonctions sont très utiles pour créer ou modifier des fichiers, écrire des logs ou des rapports, ou encore pour sauvegarder des données générées par une application.

La fonction `fwrite()` permet d'écrire des données dans un fichier, en utilisant la ressource de fichier retournée par la fonction `fopen()`. Elle prend 3 paramètres en entrée : la ressource de fichier, la chaîne de caractères à écrire et la taille de la chaîne. Cette fonction peut écrire des données en mode texte ou binaire. En mode texte, les données sont écrites directement dans le fichier sans traitement particulier. En mode binaire, les données sont écrites en format binaire structuré, par exemple pour écrire des fichiers images, audio ou des données compressées.

La fonction `fwrite()` retourne le nombre d'octets écrits dans le fichier. Ce retour est très utile pour vérifier que toutes les données ont bien été écrites dans le fichier. Si le nombre d'octets retourné est inférieur à la taille de la chaîne, cela peut indiquer une erreur lors de l'écriture des données. L'usage de la fonction `fwrite()` est très courant, par exemple on peut écrire une chaîne de caractères dans un fichier en mode texte comme ceci :

```
1 $file = fopen("monfichier.txt", "w");
2 $content = "Bonjour, comment ça va?";
3 $bytes_written = fwrite($file, $content);
4 fclose($file);
5 if($bytes_written !== false) {
6     echo "Ecriture de " . $bytes_written . " octets réussie.";
7 } else {
8     echo "Erreur lors de l'écriture du fichier.";
9 }
```

#### Exemple

Dans ce cas précis, la fonction `fopen()` est utilisée avec le mode d'écriture ("w"), qui permet de créer un nouveau fichier ou d'écraser le contenu d'un fichier existant. La fonction `fwrite()` permet d'écrire la chaîne de caractères définie dans le fichier ouvert, en utilisant la ressource de fichier obtenue grâce à la fonction `fopen()`. Le nombre d'octets écrits est stocké dans la variable `$bytes_written`.

Enfin, la fonction `fclose()` est appelée pour fermer le fichier ouvert. Cet exemple montre comment la fonction `fwrite()` peut être utilisée pour écrire des données dans un fichier en PHP.

### L'écriture - `file_put_contents()`

Une autre fonction d'écriture couramment utilisée en PHP est la fonction `file_put_contents()`. Cette fonction permet d'écrire une chaîne de caractères dans un fichier en une seule étape, sans avoir besoin d'ouvrir et de fermer explicitement le fichier. La fonction `file_put_contents()` prend 2 paramètres : le nom du fichier à écrire et la chaîne de caractères à écrire. Elle peut également prendre un troisième paramètre facultatif, qui permet de spécifier des options supplémentaires, telles que le mode d'ouverture du fichier ou le comportement en cas de collision de noms de fichiers.

Voici un exemple d'utilisation de la fonction `file_put_contents()` pour écrire une chaîne de caractères dans un fichier :

```
1 $file = 'monfichier.txt';
2 $content = 'Contenu à écrire dans le fichier';
3 file_put_contents($file, $content);
```

#### Exemple

Le code ci-dessus utilise la fonction `file_put_contents()` pour écrire le contenu de la variable `$content` dans un fichier nommé « *monfichier.txt* ». Dans cet exemple, le nom du fichier est stocké dans la variable `$file` et le contenu est stocké dans la variable `$content`. Lorsque la fonction `file_put_contents()` est appelée avec ces 2 paramètres, elle ouvre le fichier en mode écriture, écrit le contenu dans le fichier et ferme le fichier. Si le fichier n'existe pas, la fonction le crée. Si le fichier existe déjà, son contenu est écrasé par le nouveau contenu. Ainsi, on comprend que la fonction `file_put_contents()` est une fonction pratique et simple à utiliser pour écrire du contenu dans un fichier en une seule ligne de code.

#### Méthode

```
1 $file = fopen("monfichier1.txt", "w");
2 if($file) {
3     fwrite($file, " Voici un texte d'exemple pour monfichier1.txt");
4 }
5 fclose($file);
6 $count = 0;
7 $file_count = fopen('monfichier1.txt', 'r');
```

Dans cette section de code, nous commençons par créer un fichier nommé « *monfichier1.txt* » en utilisant la fonction `fopen` de PHP. Nous utilisons le mode "w" pour ouvrir le fichier en écriture, ce qui signifie que si le fichier n'existe pas, il sera créé et si le fichier existe déjà, son contenu sera écrasé. Ensuite, nous vérifions si la création du fichier a été réussie en utilisant une instruction conditionnelle `if`. Si la création du fichier a été réussie, nous écrivons un texte d'exemple en utilisant la fonction `fwrite` de PHP. Après avoir écrit le texte, nous fermons le fichier en utilisant la fonction `fclose` de PHP. Enfin, nous initialisons une variable `$count` à 0 et ouvrons le fichier « *monfichier1.txt* » en mode lecture, en utilisant la fonction `fopen`. Ceci nous permettra de compter le nombre de ligne dans le fichier en utilisant une boucle `while` et la fonction `fgets`.

```
1 while(!feof($file_count)) {
2     $line = fgets($file_count);
3     $count++; }
4 echo " Le fichier contient " . $count . " lignes";
5 $file_add = fopen('monfichier1.txt', 'a');
```

Ensuite, nous initialisons une boucle `while` qui se poursuit jusqu'à ce que la fin du fichier soit atteinte en utilisant `feof`.

#### Définition

La fonction `feof()` teste si nous avons atteint la fin du fichier. Elle prend un argument qui est le pointeur de fichier retourné par la fonction `fopen()` et renvoie « *true* » si le pointeur est à la fin du fichier, sinon elle renvoie « *false* ».

#### Méthode

À chaque tour de boucle, nous utilisons la fonction `fgets` de PHP pour lire une ligne du fichier, que nous stockons dans une variable `$line`. Ensuite, nous incrémentons la variable `$count` de 1 pour compter le nombre de lignes dans le fichier. Une fois que la boucle est terminée, nous affichons le nombre de lignes dans le fichier, à

l'aide de la fonction `echo` de PHP. Enfin, nous ouvrons le fichier « *monfichier1.txt* » en mode ajout, en utilisant la fonction `fopen` de PHP. Le mode « *a* » signifie que si le fichier existe déjà, les nouvelles données seront ajoutées à la fin du fichier, plutôt que d'écraser le contenu existant.

**Exemple**

```
1 if($file_add) {  
2     fwrite($file_add, "\nNouvelle entrée");  
3     fclose($file_add);  
4 }  
5 $file_update = fopen('monfichier1.txt', 'r');  
6 echo "<br><br> Le fichier contient désormais : ";  
7 while(!feof($file_update)) {  
8     $line = fgets($file_update);  
9     echo "<br>".$line;  
10 }  
11 fclose($file_update);
```

**Méthode**

Si l'ouverture du fichier a réussi, nous écrivons une nouvelle ligne dans le fichier en utilisant la fonction `fwrite`. Nous ajoutons ensuite une instruction `fclose` pour fermer le fichier. Ensuite, nous ouvrons le fichier « *monfichier1.txt* » en mode lecture en utilisant la fonction `fopen` et stockons le pointeur de fichier dans la variable `$file_update`. Nous affichons ensuite un message pour indiquer que nous allons afficher le contenu mis à jour du fichier. Nous initialisons ensuite une boucle `while` qui se poursuit jusqu'à ce que la fin du fichier soit atteinte en utilisant la fonction `feof`. À chaque tour de boucle, nous utilisons la fonction `fgets` pour lire une ligne du fichier, que nous stockons dans une variable `$line`. Nous affichons ensuite chaque ligne en utilisant la fonction `echo`. Enfin, nous fermons le fichier en utilisant la fonction `fclose`.

**B. Exercice : Quiz**

[solution n°1 p.17]

## Question 1

Quelle fonction permet de lire une ligne à partir d'un fichier en utilisant une ressource de fichier retournée par la fonction `fopen()` ?

- ☐ `fread()`
- ☐ `file_get_contents()`
- ☐ `fgets()`

## Question 2

Quelle fonction permet de fermer un fichier ouvert ?

- ☐ `fclose()`
- ☐ `close()`
- ☐ `endfile()`

## Question 3

Quelle fonction permet d'écrire une chaîne de caractères dans un fichier en une seule ligne de code ?

- ☐ `fwrite()`
- ☐ `file_put_contents()`
- ☐ `file_write()`

#### Question 4

Quelle fonction permet de lire un fichier en entier ?

- ☐ `file_get_contents()`
- ☐ `fgets()`
- ☐ `fread()`

#### Question 5

Quelle fonction permet d'ouvrir un fichier pour l'écriture ?

- ☐ `fopen()`
- ☐ `fgets()`
- ☐ `file_get_contents()`

### III. La manipulation de fichiers en PHP : `copy()`, `rename()`, `unlink()` et `file_exists()`

#### A. La manipulation de fichiers en PHP : `copy()`, `rename()`, `unlink()` et `file_exists()`

Dans cette seconde partie de notre cours sur la manipulation de fichiers, nous allons explorer des fonctions supplémentaires pour une gestion plus complète des fichiers. En plus des fonctions de lecture et d'écriture que nous avons déjà étudiées, il est crucial de savoir copier, renommer, supprimer et vérifier l'existence des fichiers pour une utilisation plus avancée en PHP. Ces fonctions sont indispensables pour de nombreux projets de développement web et vous permettront de gérer efficacement vos fichiers en PHP. Nous allons nous pencher sur 4 fonctions importantes pour la gestion de fichiers : `copy()`, `rename()` et `unlink()` et `file_exists()`.

#### `copy()`

Tout d'abord, la fonction `copy()` permet de créer une copie d'un fichier existant. Elle prend en entrée le chemin d'accès au fichier à copier ainsi que le chemin où la copie doit être enregistrée. Il est important de noter que la fonction `copy()` ne supprime pas le fichier original et qu'elle peut lever une exception si le fichier cible existe déjà. Pour éviter cela, il est recommandé de vérifier au préalable si le fichier existe à l'aide de la fonction `file_exists`.

#### Exemple

Voici un exemple d'utilisation de la fonction `copy()` :

```
1 <?php
2 if (!copy('source.txt', 'destination.txt')) {
3     echo "La copie du fichier a échoué";
4 }
5 ?>
```

Le code PHP que nous avons vu ci-dessus est utilisé pour copier un fichier nommé « *source.txt* » vers un autre fichier nommé « *destination.txt* ». Pour ce faire, il utilise la fonction `copy()` de PHP qui permet de copier un fichier d'un emplacement à un autre. Si la copie a échoué, le code affiche un message d'erreur.



Pour utiliser la fonction `copy()`, il faut donner 2 arguments : le nom du fichier source et le nom du fichier destination. Dans notre exemple, le premier argument est le nom du fichier source « *source.txt* » et le deuxième argument est le nom du fichier destination « *destination.txt* ». Si la copie est réussie, la fonction retourne « *true* », sinon elle retourne « *false* ».

Dans notre cas, si la copie est réussie, le code ne fera rien d'autre que de se terminer. Si la copie échoue, le code affichera le message d'erreur : « *La copie du fichier a échoué* ». Il est important de noter que le code doit être exécuté sur un serveur web pour fonctionner correctement. Il est également important que le fichier « *source.txt* » soit présent dans le même répertoire que le fichier PHP.

#### Exemple

Voici un exemple d'utilisation de la fonction `copy()` pour copier un fichier nommé « *image.jpg* » dans le dossier « *images* » :

```
1 if (copy('image.jpg', 'images/image.jpg')) {
2     echo "Le fichier a été copié avec succès";
3 } else {
4     echo "La copie du fichier a échoué";
5 }
```

Dans cet exemple, la fonction `copy()` copie le fichier « *image.jpg* » dans le dossier « *images* ». Si la copie est réussie, le code affiche le message « *Le fichier a été copié avec succès* ». Si la copie échoue, le code affiche le message d'erreur. Passons maintenant à la fonction `rename()` en PHP, qui permet de renommer un fichier ou de le déplacer vers un autre emplacement.

### rename()

La fonction `rename()` permet de renommer un fichier ou de le déplacer dans un autre dossier. Elle prend en entrée le chemin d'accès au fichier à renommer ainsi que le nouveau nom ou le nouveau chemin où le fichier doit être déplacé. Si le fichier cible existe déjà, la fonction `rename()` le remplacera. Il est donc nécessaire d'être prudent lors de l'utilisation de cette fonction et de vérifier au préalable si le fichier cible existe à l'aide de la fonction `file_exists()`.

#### Exemple

Voici un exemple d'utilisation de la fonction `rename()` :

```
1 <?php
2     if (!rename('ancien_nom.txt', 'nouveau_nom.txt')) {
3         echo "Operation échoué";
4     }
5 ?>
```

Le code PHP présenté ci-dessus démontre comment la fonction `rename()` peut être utilisée pour renommer un fichier dans un script PHP. Cette fonction nécessite 2 arguments : le nom actuel du fichier et le nouveau nom que vous souhaitez lui donner. Dans l'exemple, la fonction `rename()` est utilisée pour renommer le fichier « *ancien\_nom.txt* » en « *nouveau\_nom.txt* ». Il convient de noter que la fonction `rename()` ne peut renommer un fichier que s'il se trouve dans le même répertoire que le script PHP qui l'exécute. Par ailleurs, l'utilisateur qui exécute le script doit disposer des **autorisations nécessaires** pour renommer le fichier.

### unlink()

Enfin, la fonction `unlink()` permet de supprimer un fichier. Elle prend en entrée le chemin d'accès au fichier à supprimer et supprime définitivement ce fichier. Il est important de noter que la fonction `unlink()` peut lever une exception si le fichier n'existe pas ou si l'utilisateur n'a pas les permissions nécessaires pour le supprimer.

### Exemple

Voici un exemple de code pour illustrer :

```
1 <?php
2 if (!unlink('nom_du_fichier.txt')) {
3     echo "La suppression du fichier a échoué";
4 }
5 ?>
```

Le code ci-dessus illustre l'utilisation de la fonction `unlink()` en PHP pour supprimer un fichier. La fonction prend en argument le chemin du fichier à supprimer. Cependant, il est important de noter que la fonction `unlink()` ne peut supprimer un fichier que s'il se trouve dans le même répertoire que le fichier PHP qui exécute la fonction. De plus, l'utilisateur doit avoir les permissions nécessaires pour supprimer le fichier. Cette fonction est pratique pour nettoyer l'espace de stockage en supprimant des fichiers inutiles ou obsolètes. Cependant, il est important de l'utiliser avec précaution, car elle supprime définitivement le fichier et ne peut pas être annulée.

En conclusion, les fonctions `copy()`, `rename()`, `unlink()` et `file_exists()` sont des outils indispensables pour la manipulation de fichiers en programmation. Il convient cependant d'être vigilant lors de leur utilisation et de vérifier si le fichier cible existe déjà avant de procéder à une copie, un renommage ou une suppression pour éviter toute perte de données. La fonction `file_exists()` permet justement de vérifier l'existence d'un fichier avant de procéder à une opération de manipulation de fichiers, en complément des fonctions `copy()`, `rename()`, `unlink()`.

Il est crucial de vérifier l'existence d'un fichier afin d'éviter les erreurs et les pertes de données. Cette fonction prend en argument le nom du fichier à vérifier et renvoie « *true* » si le fichier existe, ou « *false* » s'il n'existe pas.

### Exemple

Voici un exemple illustrant l'utilisation de la fonction `file_exists()` :

```
1 <?php
2 $file = "mon_fichier.txt";
3 if (file_exists($file)) {
4     echo "Le fichier $file existe.";
5 } else {
6     echo "Le fichier $file n'existe pas.";
7 }
8 ?>
```

Ce code est un exemple simple de l'utilisation de la fonction `file_exists()` pour vérifier l'existence d'un fichier avant de l'opérer.

### Exemple

```
1 <?php
2 if ($_SERVER['REQUEST_METHOD'] === "POST") {
3     $file = $_FILES['file']['name'];
4     $destination = "uploads/";
5     $target_dir = $destination . basename($file);
```

Dans cette partie de code, nous avons notre variable `$file` qui contient le nom du fichier qui a été téléchargé, et une variable `$destination` qui représente le répertoire où nous voulons enregistrer le fichier. La variable `$target_dir` est la variable importante, car elle contient le chemin complet où le fichier sera enregistré, c'est-à-dire le répertoire de destination plus le nom du fichier.

```

1 if (file_exists($target_dir)) {
2     $backup_file = $destination . 'backup' . basename($file);
3     copy($target_dir, $backup_file);
4     $rename_file = $destination . 'rename' . basename($file);
5     rename($target_dir, $rename_file);
6     unlink($backup_file);
7     echo "Le fichier a bien été remplacé avec succès";
8 }

```

Ce code vérifie si le fichier que vous voulez télécharger possède le même nom qu'un fichier qui existe déjà dans le dossier où il doit être enregistré. S'il existe, il crée une copie de sauvegarde du fichier, puis renomme le fichier original en ajoutant le mot « *rename* » devant son nom, pour éviter de perdre les données qu'il contient. Enfin, il supprime le fichier renommé pour que le fichier téléchargé puisse être enregistré à sa place. Ces étapes permettent de protéger le fichier téléchargé et de s'assurer qu'il n'écrase pas accidentellement un fichier existant qui pourrait être important.

```

1 if(move_uploaded_file($_FILES['file']['tmp_name'], $target_dir) {
2     echo "Le fichier ".$file." a bien été téléchargé";
3 } else {
4     echo "Il y a eu une erreur lors du téléchargement du fichier";
5 }

```

Ce code s'assure que si le fichier n'est pas déjà présent dans le répertoire de destination, il sera téléchargé avec succès. Mais si une erreur se produit pendant le processus de téléchargement, le code informera l'utilisateur de l'échec du téléchargement.

En résumé, ces exemples ont permis de découvrir en action : `file_exists()`, `copy()`, `rename()` et `unlink()`. Nous avons appris comment vérifier si un fichier existe déjà dans un répertoire de destination et comment créer une sauvegarde avant de le remplacer. De plus, nous avons vu comment renommer des fichiers et comment télécharger des fichiers en utilisant une méthode `POST`. Ces fonctions sont indispensables pour toute opération de manipulation de fichiers et constituent des outils clés.

### La manipulation des répertoires de fichiers : `is_dir()` et `mkdir()`

En plus des 4 fonctions que nous avons déjà vues, il existe d'autres fonctions utiles qui nous permettront d'élargir nos compétences en travaillant avec les répertoires de fichiers. La fonction `is_dir()` peut être utilisée pour vérifier si un fichier est un répertoire ou non. La fonction `mkdir()` permet de créer un nouveau répertoire dans le système de fichiers, tandis que la fonction `rmdir()` peut être utilisée pour supprimer un répertoire vide. La fonction `is_dir()` prend un paramètre qui représente le chemin du fichier à vérifier et renvoie « *true* » si le fichier est un répertoire et « *false* » s'il ne l'est pas. La fonction `mkdir()` prend également un paramètre qui représente le chemin du nouveau répertoire à créer et peut également prendre un deuxième paramètre facultatif qui représente les options pour la création du répertoire. La fonction `rmdir()` est utilisée pour supprimer un répertoire vide. Elle prend un seul paramètre qui représente le chemin du répertoire à supprimer. Il est important de noter que cette fonction ne peut supprimer que des répertoires vides. Si vous essayez de supprimer un répertoire qui contient des fichiers, vous recevrez une erreur.

#### Exemple

```

1 if(is_dir('documents')) {
2     echo " Le répertoire 'documents' existe déjà<br> ";
3 }else {
4     if(mkdir('documents')){
5         echo " Le répertoire 'documents' a bien été crée <br>";
6     }
7 }

```

Ce code PHP permet de vérifier si le répertoire « *documents* » existe. Si oui, il affiche un message indiquant que le répertoire existe déjà. Sinon, il crée un nouveau répertoire nommé « *documents* » à l'aide de la fonction `mkdir()` et affiche un message confirmant la création du répertoire. La fonction `mkdir()` renvoie « *true* » si le répertoire est créé avec succès, sinon elle renvoie « *false* » en cas d'erreur.

#### Exemple

```
1 $file = fopen('documents/monfichier.txt', 'w');
2 if($file) {
3     fwrite($file, " Voici un texte d'exemple");
4     fclose($file);
5 } else
6 {
7     echo " Impossible de créer le fichier dans le répertoire 'documents'";
8 }
```

Dans ce bout de code PHP, on peut voir l'ouverture d'un fichier nommé « *monfichier.txt* » en mode écriture (*w*) dans le répertoire « *documents* ». Si l'ouverture du fichier est réussie, on utilise la fonction `fwrite()` pour y écrire le texte « *Voici un texte d'exemple* ». Puis, on referme le fichier grâce à la fonction `fclose()`. Si l'ouverture du fichier échoue pour une raison quelconque, un message d'erreur est affiché pour signaler que le fichier n'a pas pu être créé dans le répertoire « *documents* ».

#### Exemple

```
1 <?php
2 if (file_exists('documents/monfichier.txt')) {
3     if (is_dir('backup')) {
4         echo "Le répertoire 'backup' existe déjà <br>";
5     } else {
6         mkdir('backup');
7     }
8     if (copy('documents/monfichier.txt', 'backup/monfichier.txt')) {
9         echo "Le fichier 'monfichier.txt' a bien été copié dans le répertoire 'backup' <br>";
10    } else {
11        echo "Échec de la copie du fichier 'monfichier.txt' <br>";
12    }
13 } else {
14     echo "Le fichier 'monfichier.txt' n'existe pas dans le répertoire 'documents' <br>";
15 }
16 ?>
```

Pour finir, ce code commence par vérifier si le fichier « *monfichier.txt* » existe déjà dans le répertoire « *documents* ». S'il existe, il crée un nouveau répertoire nommé « *backup* » en utilisant la fonction `mkdir()` et vérifie s'il existe déjà. Si le répertoire existe déjà, un message est affiché pour le signaler. Sinon, la fonction `copy()` est utilisée pour copier le fichier « *monfichier.txt* » dans le répertoire « *backup* » nouvellement créé. Si la copie est réussie, un message est affiché pour indiquer que le fichier a bien été copié dans le répertoire « *backup* ». Notez que la fonction `copy()` renvoie « *true* » si la copie réussit et « *false* » si elle échoue.

## B. Exercice : Quiz

[solution n°2 p.18]

### Question 1

La fonction `copy()` permet de créer une copie de fichier.

- ☐ Vrai
- ☐ Faux

## Question 2

Quelle fonction permet de renommer un fichier ?

- ☐ `rename()`
- ☐ `rewrite()`
- ☐ `changenname()`

## Question 4

Quelle fonction permet de créer un nouveau répertoire dans le système de fichiers ?

- ☐ `is_dir()`
- ☐ `mkdir()`
- ☐ `rmdir()`

## Question 5

Quelle fonction permet d'écrire du texte dans un fichier ?

- ☐ `rewrite()`
- ☐ `fwrite()`
- ☐ `fclose()`

## IV. Essentiel

En conclusion, ce cours nous a permis de découvrir l'importance de la manipulation de fichiers en PHP et les nombreuses fonctionnalités proposées par ce langage pour y faire face. La manipulation de fichiers est une compétence essentielle pour tout développeur web, et la maîtrise de ces techniques en PHP peut être très utile pour simplifier les tâches répétitives et améliorer la performance de notre code. En plus de la lecture de fichiers, nous avons également abordé la création, l'ouverture et la fermeture de fichiers, ainsi que les fonctions pour renommer, copier, supprimer et vérifier l'existence de fichiers, mais aussi les opérations de manipulation des répertoires. Ces dernières sont particulièrement importantes pour organiser les fichiers en groupes logiques et naviguer dans les fichiers d'un site web. En résumé, la manipulation de fichiers est une compétence cruciale pour la création et la gestion de sites web dynamiques en PHP. Tout développeur PHP doit donc maîtriser ces fonctions de base pour gérer efficacement les fichiers et garantir le bon fonctionnement des sites web.

## V. Auto-évaluation

### A. Exercice

Vous êtes responsable du développement d'une application web pour une entreprise de vente en ligne.

#### Question 1

[solution n°3 p.19]

À partir du formulaire fourni, pouvez-vous écrire un script PHP qui permette de stocker les informations de commande dans un fichier texte et de créer une copie de sauvegarde de ce fichier dans un dossier spécifié ?

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Formulaire de commande</title>
5   </head>
6   <body>
7     <h1>Formulaire de commande</h1>
8     <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

```

9      <label for="nom">Nom :</label>
10     <input type="text" name="nom" id="nom" required><br><br>
11     <label for="adresse">Adresse :</label>
12     <textarea name="adresse" id="adresse" required></textarea><br><br>
13     <label for="produit">Produit :</label>
14     <input type="text" name="produit" id="produit" required><br><br>
15     <label for="prix">Prix :</label>
16     <input type="number" name="prix" id="prix" required><br><br>
17     <input type="submit" value="Envoyer">
18 </form>
19 </body>
20 </html>

```

## Question 2

[solution n°4 p.19]

Comment pouvez-vous ajouter une fonctionnalité pour permettre la lecture des commandes à partir du fichier de stockage ?

## B. Test

### Exercice 1 : Quiz

[solution n°5 p.19]

#### Question 1

Quelle fonction permet d'ouvrir un fichier en mode lecture seule ?

- ☐ `fopen('nom_fichier', 'r+')`
- ☐ `fopen('nom_fichier', 'w');`
- ☐ `fopen('nom_fichier', 'r');`

#### Question 2

Quelle fonction permet de lire le contenu d'un fichier en entier et de le stocker dans une variable ?

- ☐ `fread();`
- ☐ `fgets();`
- ☐ `file_get_contents();`

#### Question 3

Quelle fonction permet de vérifier si un fichier existe avant de le manipuler ?

- ☐ `file_exists();`
- ☐ `is_file();`
- ☐ Les deux

#### Question 4

Quelle fonction permet d'écrire dans un fichier en mode ajout à la fin ?

- ☐ `fopen('nom_fichier', 'w');`
- ☐ `fopen('nom_fichier', 'a');`
- ☐ `fopen('nom_fichier', 'r+');`

#### Question 5

Quelle fonction permet de supprimer un fichier ?

- ☐ `delete_file();`
- ☐ `unlink();`
- ☐ `remove_file();`

## Solutions des exercices






**Exercice p. 7 Solution n°1****Question 1**

Quelle fonction permet de lire une ligne à partir d'un fichier en utilisant une ressource de fichier retournée par la fonction `fopen()` ?


- ☐ `fread()`
- ☐ `file_get_contents()`
- ☒ `fgets()`

 En utilisant la ressource de fichier renvoyée par la fonction `fopen()`, la fonction `fgets()` permet de lire une ligne à partir d'un fichier.

**Question 2**

Quelle fonction permet de fermer un fichier ouvert ?


- ☒ `fclose()`
- ☐ `close()`
- ☐ `endfile()`

 En PHP, la fonction `fclose()` est utilisée pour fermer un fichier qui a été ouvert précédemment à l'aide de la fonction `fopen()`.

**Question 3**

Quelle fonction permet d'écrire une chaîne de caractères dans un fichier en une seule ligne de code ?


- ☐ `fwrite()`
- ☒ `file_put_contents()`
- ☐ `file_write()`

 En utilisant la fonction `file_put_contents()`, il est possible d'insérer facilement une chaîne de caractères dans un fichier en une seule ligne de code.

**Question 4**


Quelle fonction permet de lire un fichier en entier ?

- ☒ `file_get_contents()`
- ☐ `fgets()`
- ☐ `fread()`

 La fonction `file_get_contents()` permet de récupérer tout le contenu d'un fichier et de le renvoyer sous forme de chaîne de caractères.

**Question 5**


Quelle fonction permet d'ouvrir un fichier pour l'écriture ?

- ☒ `fopen()`
- ☐ `fgets()`
- ☐ `file_get_contents()`
-  La fonction `fopen()` permet d'ouvrir un fichier avec différents modes, notamment en lecture seule ("`r`"), en écriture ("`w`") ou en ajoutant du contenu ("`a`").

## Exercice p. 12 Solution n°2


### Question 1

La fonction `copy()` permet de créer une copie de fichier.

- ☒ Vrai
- ☐ Faux
-  La fonction `copy()` de PHP permet de copier un fichier d'un emplacement à un autre.


### Question 2

Quelle fonction permet de renommer un fichier ?

- ☒ `rename()`
- ☐ `rewrite()`
- ☐ `changenam()`
-  C'est la fonction `rename()` qui permet de renommer un fichier.


### Question 4

Quelle fonction permet de créer un nouveau répertoire dans le système de fichiers ?

- ☐ `is_dir()`
- ☒ `mkdir()`
- ☐ `rmdir()`
-  La fonction `mkdir()` permet de créer un nouveau répertoire dans le système de fichiers, la fonction `rmdir()` permet de supprimer un répertoire vide. La fonction `is_dir()` permet de vérifier si le fichier est un répertoire.

### Question 5

Quelle fonction permet d'écrire du texte dans un fichier ?

- ☐ `rewrite()`
- ☒ `fwrite()`
- ☐ `fclose()`
-  La fonction `fwrite()` permet d'écrire du texte dans un fichier.

## p. 13 Solution n°3

```

1 <?php
2 If ($_SERVER['REQUEST_METHOD'] == 'POST') {
3     If (isset($_POST['nom'])) {
4         $nom_client = $_POST['nom'];
5         $adresse_client = $_POST['adresse'];
6         $produit_commande = $_POST['produit'];
7         $prix_commande = $_POST['prix'];
8         $file = fopen('commandes.txt', 'a');
9         fwrite($file, "$nom_client, $adresse_client, $produit_commande, $prix_commande\n");
10        fclose($file);
11        echo "La commande a été enregistrée avec succès !<br>";
12        if (!is_dir('backup')) {
13            mkdir('backup');
14        }
15        copy('commandes.txt', 'backup/commandes_backup.txt');
16        echo "La commande a été sauvegardée avec succès !";
17    }
18    $file = fopen('commandes.txt', 'r');

```

## p. 14 Solution n°4

```

1 $file = fopen('commandes.txt', 'r');
2 while(!feof($file)) {
3     $line = fgets($file);
4     echo "<br>".$line;
5 }
6 fclose($file);


```

## Exercice p. 14 Solution n°5

## Question 1

Quelle fonction permet d'ouvrir un fichier en mode lecture seule ?


- ☐ fopen('nom\_fichier', 'r+')
- ☐ fopen('nom\_fichier', 'w');
- ☒ fopen('nom\_fichier', 'r');

 La fonction `fopen()` permet d'ouvrir un fichier. En utilisant le mode 'r', le fichier est ouvert en lecture seule.

## Question 2

Quelle fonction permet de lire le contenu d'un fichier en entier et de le stocker dans une variable ?


- ☐ `fread()`;
- ☐ `fgets()`;
- ☒ `file_get_contents()`;

 La fonction `file_get_contents()` permet de lire le contenu d'un fichier en entier et de le stocker dans une variable.

### Question 3

Quelle fonction permet de vérifier si un fichier existe avant de le manipuler ?


- ☐ `file_exists()` ;
- ☐ `is_file()` ;
- ☒ Les deux

 Les fonctions `file_exists()` et `is_file()` permettent toutes les deux de vérifier si un fichier existe avant de le manipuler. Néanmoins, `file_exists()` vérifie l'existence d'un fichier ou d'un répertoire, tandis que `is_file()` vérifie si le fichier spécifié est un fichier normal.

### Question 4

Quelle fonction permet d'écrire dans un fichier en mode ajout à la fin ?


- ☐ `fopen('nom_fichier', 'w');`
- ☒ `fopen('nom_fichier', 'a');`
- ☐ `fopen('nom_fichier', 'r+');`

 La fonction `fopen()` permet d'ouvrir un fichier en utilisant le mode 'a' pour écrire dans un fichier en mode ajout à la fin.

### Question 5

Quelle fonction permet de supprimer un fichier ?

- ☐ `delete_file()` ;
- ☒ `unlink()` ;
- ☐ `remove_file()` ;

 La fonction `unlink()` permet de supprimer un fichier.