



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fichiers](#) ► [Inclusion de fichiers](#)

Inclusion de fichiers

PHP nous permet d'inclure un fichier dans un script php. Mais pourquoi inclure un fichier dans un autre ? Lorsque nos programmes commenceront à acquérir une taille respectable, il sera très pratique d'isoler des parties du code dans des fichiers, puis d'inclure ces fichiers lorsque nous en aurons besoin. Par exemple, il est d'usage d'isoler les fonctions selon leurs utilités dans un ou plusieurs fichiers, et d'inclure ce fichier dans le script qui appelle ces fonctions. On constitue ainsi des bibliothèques de fonctions que l'on peut utiliser dans n'importe quel script une fois la bibliothèque incluse.

Autre avantage non négligeable, si je veux utiliser une même fonction dans plusieurs scripts, je déclare cette fonction dans un fichier que j'inclus ensuite dans les différents scripts ; la fonction n'a été écrite qu'une seule fois. Autre possibilité : dans une page web il sera très pratique d'isoler des "morceaux" de la page dans des scripts plus petits pour enfin les réunir dans la page maîtresse avec des inclusions. Par exemple je crée mon en-tête dans un fichier header.html, le corps de ma page dans un fichier content.php et le pied de page dans un fichier footer.php. Ensuite je n'ai plus qu'à inclure ces trois fichiers dans le fichier la page mapage.php. Les fichiers inclus seront très souvent des fichiers de script php ou des fichiers html.

Les instructions include et require permettent d'inclure un fichier dans un autre. Le chemin relatif ou absolu vers le fichier doit être "passé en argument". La différence entre include et require tient à l'erreur renvoyée en cas de problème : include renverra un WARNING tandis que require renverra une E_COMPILE_ERROR.

Nous créons un script qui déclare une variable nb et lui assigne la valeur 456 puis qui affiche la valeur de cette variable. Puis nous incluons ce script dans un autre qui ajoute à nb la valeur 100 et qui affiche la nouvelle valeur.

```
1 <?php
2
3 // inclusion_fichiers_1.php
4 $nb = 456;
5 echo "\$nb a pour valeur $nb" . PHP_EOL;
```

```
1 <?php
2
3 // inclusion_fichiers_2.php
4 include './inclusion_fichiers_1.php';
5 $nb += 100;
6 echo "maintenant \$nb a pour valeur $nb" . PHP_EOL;
```

L'exécution du fichier inclusion_fichiers_2.php affiche :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php inclusion_fichiers_2.php
$nb a pour valeur 456
maintenant $nb a pour valeur 556
patrice@patrice-laptop:~/PhpProject3$
```



La variable `nb` a été déclarée dans `inclusion_fichiers_1.php` mais puisque ce fichier est inclus dans `inclusion_fichiers_2.php`, cette variable est utilisable dans le script `inclusion_fichiers_2.php`. Lors de l'inclusion les instructions qui sont écrites dans le fichier inclus sont exécutées comme si elles faisaient parties du fichier incluant.

Un autre exemple avec un fichiers de fonctions inclus dans un autre fichier :

```
1 <?php
2
3 // inclusion_fichiers_3.php
4 function additionner($nb1, $nb2) {
5     return $nb1 + $nb2;
6 }
7
8 function soustraire($nb1, $nb2) {
9     return $nb1 - $nb2;
10 }
11
12 function multiplier($nb1, $nb2) {
13     return $nb1 * $nb2;
14 }
```

```
1 <?php
2
3 // inclusion_fichiers_4.php
4 include './inclusion_fichiers_3.php';
5 $nb1 = 3;
6 $nb2 = 5;
7 echo additionner($nb1, $nb2) . PHP_EOL;
8 echo soustraire($nb1, $nb2) . PHP_EOL;
9 echo multiplier($nb1, $nb2) . PHP_EOL;
```

Les instructions `include_once` et `require_once` font le même travail que `include` et `require`, en renvoyant les même erreurs en cas de problème, mais elles n'incluent qu'une seule fois le script. Par exemple :

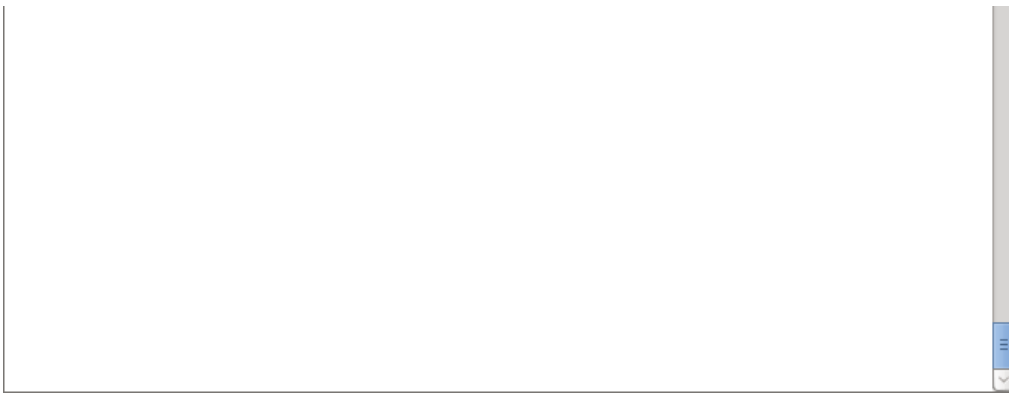
```
1 <?php
2
3 // inclusion_fichiers_5.php
4 echo 'Hello !' . PHP_EOL;
```

Avec des `include`

```
1 <?php
2
3 // inclusion_fichiers_6.php
4 include './inclusion_fichiers_5.php';
5 include './inclusion_fichiers_5.php';
```

L'exécution affiche :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php inclusion_fichiers_6.php
Hello !
Hello !
patrice@patrice-laptop:~/PhpProject3$
```



Avec des `include_once` :

```
1 <?php
2
3 // inclusion_fichiers_7.php
4 include_once './inclusion_fichiers_5.php';
5 include_once './inclusion_fichiers_5.php';
```

L'exécution affiche cette fois :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php inclusion_fichiers_7.php
Hello !
patrice@patrice-laptop:~/PhpProject3$
```

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 structure 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](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fichiers](#) ► [Lecture rapide de fichier](#)

Lecture rapide de fichier

Plusieurs fonctions PHP permettent de lire "rapidement" un fichier. La fonction `file_get_contents()` : renvoie le contenu du fichier ou false si une erreur est survenue. Le chemin vers le fichier peut être absolu ou relatif.

haiku.txt

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

```
1 <?php
2
3 // lecture_rapide_fichier_1.php
4 $text = file_get_contents('haiku.txt');
5 echo $text;
```

La fonction `readfile()` affiche directement le contenu du fichier et retourne le nombre d'octets lus ou false si une erreur est survenue.

```
1 <?php
2
3 // lecture_rapide_fichier_2.php
4 readfile('haiku.txt');
```

Enfin la fonction `file()` permet de renvoyer le contenu du fichier dans un tableau ligne à ligne.

```
1 <?php
2
3 // lecture_rapide_fichier_3.php
4 $tab = file('haiku.txt');
5 print_r($tab);
```

Ces fonctions permettent aussi de récupérer le contenu d'une page web, et, éventuellement, de définir des paramètres de la requête HTTP avant l'envoi de cette dernière.

```
1 <?php
2
3 // lecture_rapide_fichier_4.php
4 $tab = file('https://www.startpage.com/fra/');
5 print_r($tab);
```

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 structure 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](#)



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fichiers](#) ► [Écriture rapide de fichier](#)

Écriture rapide de fichier

La fonction `file_put_contents()` écrit une chaîne dans un fichier. Si le fichier existe déjà il est écrasé sauf si la constante `FILE_APPEND` est passée en argument.

```
1 <?php
2
3 // ecriture_rapide_fichier_1.php
4 $data = <<<END
5 Une pierre pour oreiller
6 J'accompagne les nuages.
7 Santoka
8 END;
9 file_put_contents('autre_haiku.txt', $data);
```

Ce script écrit une chaîne dans le fichier "autre_haiku.txt", ce fichier est créé s'il n'existe pas ou écrasé s'il existe. Puis le script lit le fichier et affiche son contenu.

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 structure 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](#)



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 :

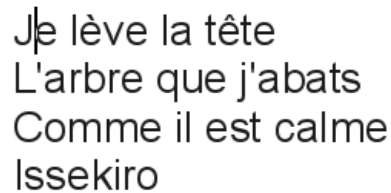
```
Je lève la tête
L'arbre que j'abats
Comme il est calme
Issekiro
```

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](#)



Introduction au langage de programmation PHP

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

Opérations sur les répertoires

À l'image de la fonction `fopen()`, la fonction `opendir()` renvoie un descripteur sur un répertoire. À partir de ce descripteur, il est possible de parcourir l'ensemble des éléments (répertoires ou fichiers) contenus dans ce répertoire avec la fonction `readdir()`. La fonction `closedir()` permet de fermer le flux et de libérer la ressource. La fonction `scandir()`, elle, renvoie le contenu du répertoire sous forme d'un tableau.

```
1 <?php
2
3 // operations_repertoires_1.php
4 $dirname = '/home/patrice';
5 $dd = opendir($dirname);
6 while (($item = readdir($dd)) !== false) {
7     $item_name = $dirname . DIRECTORY_SEPARATOR . $item;
8     echo "$item_name => " . (is_dir($item_name) ? 'répertoire' : 'fichier')
9     . PHP_EOL;
10 }
11 closedir($dd);
```

Le fonction `getcwd()` renvoie le nom du répertoire courant. La fonction `chdir()` permet de changer le répertoire courant. Enfin la fonction `rmdir()` permet d'effacer un répertoire.

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 structure 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](#)

Introduction au langage de programmation PHP

Accueil ▶ Mes cours ▶ Développement logiciel ▶ Intro PHP ▶ Les fichiers ▶ Les constantes magiques

Les constantes magiques

PHP possède un certain nombre de constantes dont la valeur dépend de l'endroit où elles sont utilisées : ce sont [les constantes magiques](#). Parmi ces constantes nous trouvons (extrait de la documentation) :

__LINE__	La ligne courante dans le fichier.
__FILE__	Le chemin complet et le nom du fichier courant. Si utilisée pour une inclusion, le nom du fichier inclus est retourné. __FILE__ contient toujours le chemin absolu pour les liens symboliques.
__DIR__	Le dossier du fichier. Si utilisée dans une inclusion, le dossier du fichier inclus sera retourné. C'est l'équivalent de dirname(__FILE__). Ce nom de dossier ne contiendra pas de slash final, sauf si c'est le dossier racine. (Ajouté en PHP 5.3.0.)

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 structure 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](#)



Introduction au langage de programmation PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [Intro PHP](#) ▶ [Les fichiers](#) ▶ [T.P. fonction copier\(\)](#)

T.P. fonction copier()

Implémentez une fonction `copier()` qui effectuera la copie d'un fichier passé en argument. Cette fonction possédera deux paramètres de type string : le premier sera le fichier à copier (la source), le deuxième sera le fichier de destination. Si le fichier de destination n'existe pas il sera créé, sinon son contenu sera écrasé. Si le fichier source et le fichier destination ne portent pas la même extension alors une erreur fatale sera déclenchée avec pour message : "extension invalide".

Le nom du script sera de la forme "fonction_copier_*nom_prenom*.php". Vous remettrez une archive zip contenant ce fichier. Le nom de l'archive sera de la forme "fonction_copier_*nom_prenom*.zip".

État du travail remis

Numéro de tentative	Ceci est la tentative 1.
Statut des travaux remis	Aucune tentative
Statut de l'évaluation	Pas évalué
Dernière modification	vendredi 27 mars 2015, 14:16

Ajouter un travail

Modifier votre travail remis

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 structure 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](#)