Les types structurés 7

PLAN

OBJECTIFS

- Maîtriser les données structurées, en particulier les tableaux et leurs traitements associés.

7.1 Introduction

Contrairement aux données simples, les variables de types structurés sont constituées de plusieurs éléments. Ceux-ci peuvent être du même type comme les chaînes de caractères ou de types différents comme les tableaux.

Elles contiennent des données simples ou structurées. Cet emboîtement de structures permet de gérer des données complexes, mais leur gestion consiste souvent à traiter les données simples internes à ces structures.

Les instructions répétitives abordées au chapitre 6 sont indispensables à la gestion des données structurées comme les tableaux ou les fichiers.

7.2 Les chaînes de caractères

Structure d'une chaîne de caractères

Une chaîne de caractères est un tableau d'octets accompagné d'un entier indiquant la longueur de la chaîne. Chaque case contient un octet, donc un caractère lorsque la table de codage numérote les caractères de 0 à 255 comme la table ASCII ou Iso-Latin1 (figure 7.1). Avec UTF-8, les caractères sur 2, 3 ou 4 octets seront mémorisés sur 2, 3 ou 4 cases du tableau.

Les différentes tables de codage des caractères, ASCII, Iso-Latin1 ou UTF-8 sont présentées à la section 5.4 du chapitre 5.

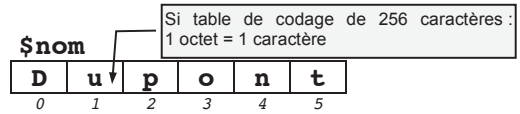


Figure 7.1 – Format d'une chaîne de caractères.

Constantes chaînes de caractères

Notation

Les guillemets `"`, ou les apostrophes `'`, définissent une constante chaîne de caractères. En voici deux exemples :

```
$texte1 = "Ceci est un texte" ;
$texte2 = 'Ceci est un texte' ;
```

Les guillemets

Les guillemets évaluent la chaîne de caractères. Si elle contient une variable, sa valeur est utilisée avant d'affecter le résultat à la variable chaîne. Dans l'exemple suivant, la variable `$message` contient le texte : `La variable i=10`.

```
$i=10 ;
$message = "La variable i=$i" ;
```

Les apostrophes

Les apostrophes n'interprètent pas le contenu. Dans l'exemple suivant, la variable `$message` contient le texte : `La variable i=$i`.

```
$i=10 ;
$message = 'La variable i=$i' ;
```

Saisie et affichage

Dans un environnement shell

La saisie et l'affichage dans un environnement shell ont déjà été présentés à la section 3.2 du chapitre 3. Le format est `%s` pour `printf()` ou `fscanf()`. La fonction `fgets(STDIN)` retourne la chaîne saisie en lisant la ligne complète (y compris la fin de ligne), la fonction `fputs(STDOUT)` affiche la chaîne de caractères.

Le programme `chaine_saisie_affichage_shell.php` saisit trois chaînes de caractères avec `fscanf()` et `fgets()` et les affiche *via* `echo`, `fprintf` et `fputs`. La fonction `trim()` supprime les espaces en début et fin de chaîne ainsi que la fin de ligne qui est saisie avec `fgets(STDIN)`.

Listing 7.1 – Programme chaine_saisie_affichage_shell.php

```
<?php
 echo "Entrez une première chaîne avec des espaces :  ";
 fscanf(STDIN,"%s",$Saisie1)                             ;
 echo "echo : Première chaîne : ".$Saisie1.PHP_EOL      ;
 echo "Entrez une deuxième chaîne sans espaces      : ";
 fscanf(STDIN,"%s",$Saisie2)                             ;
 fprintf(STDOUT,"fprintf : %s%s",$Saisie2,PHP_EOL)       ;
 echo "Entrez une troisième chaîne avec des espaces : ";
 $Saisie3=fgets(STDIN)                                   ;
 // Suppression des espaces (début,fin) et du saut de ligne
 $Saisie3=trim($Saisie3)                                 ;
 fputs(STDOUT,"fputs : ".$Saisie3.PHP_EOL)               ;
?>
```

`fscanf()` lit un seul mot. `fgets()` lit toute la ligne, y compris les espaces en début et fin. L'usage de `fgets()` impose le « nettoyage » de la chaîne lue avec la fonction `trim()`.

Dans un environnement web

- *Le formulaire*

La saisie par formulaire est abordée à la section 3.3 du chapitre 3. Le formulaire retourne une chaîne de caractères quand le type indiqué est « text ». Le fichier `chaine_saisie_affichage_web.html` en présente la saisie.

Listing 7.2 – Fichier chaine_saisie_affichage_web.html

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
   <meta charset="utf-8" />
   <title>Saisie</title>
 </head>
 <body>
  <form action="chaine_saisie_affichage_web.php" method="post">
   Entrez un nom  <input type="text" name="Nom" size="30" /><br/>
   <input type="submit" name="valider" value="Valider la saisie" />
   <input type="reset" name="effacer" value="Effacer le formulaire" />
  </form>
 </body>
</html>
```

- *Le programme PHP*

Le programme PHP `chaine_saisie_affichage_web.php` est activé par le formulaire après validation. Il récupère la donnée transmise *via* la méthode POST.

Listing 7.3 – Programme chaine_saisie_affichage_web.php

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
   <meta charset="utf-8" />
   <title>Saisie</title>
 </head>
 <body>
 <?php
 define("WEB_EOL","<br/>");
 $Nom=$_POST['Nom'] ;               // On récupère les données
 echo "Nom saisi : ".$Nom.WEB_EOL ; // On traite les données
 ?>
 </body>
</html>
```

- *Formulaire et programme PHP en une seule page*

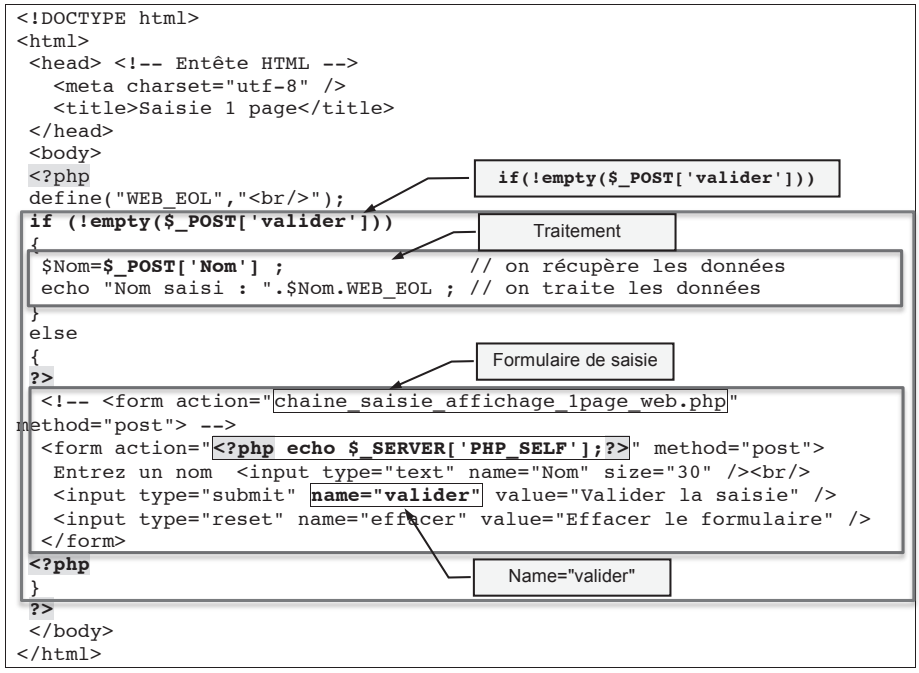
Il est possible d'écrire un unique programme PHP contenant à la fois le formulaire et le traitement de la donnée envoyée par le formulaire. Dans ce cas, le formulaire contenu dans ce programme doit appeler… le programme lui-même.

Il faut concevoir le programme pour détecter le contexte d'appel :

- le programme est appelé la première fois : le formulaire doit être affiché ;
- le programme est appelé *via* le formulaire : la donnée doit être traitée.

Un `if…else` traite ces deux cas. Le test vérifie l'existence d'une variable transmise par le formulaire. Si elle n'existe pas (vide), c'est la première exécution du programme, on affiche le formulaire. Si elle existe (non vide), le programme est donc appelé *via* le formulaire, on traite la donnée.

Dans le programme `chaine_saisie_affichage_1page_web.php` présenté à la figure 7.2, le formulaire appelle le programme lui-même et nomme l'action de validation « valider ». Le test vérifie si `$_POST['valider']` est non vide. Si c'est le cas, on traite la donnée, sinon on affiche le formulaire.




```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
   <meta charset="utf-8" />
   <title>Saisie 1 page</title>
 </head>
 <body>
 <?php
 define("WEB_EOL","<br/>");
 if (!empty($_POST['valider']))
 {
  $Nom=$_POST['Nom'] ;                // on récupère les données
  echo "Nom saisi : ".$Nom.WEB_EOL ; // on traite les données
 }
 else
 {
 ?>
  <!-- <form action="chaine_saisie_affichage_1page_web.php"
method="post"> -->
  <form action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
   Entrez un nom  <input type="text" name="Nom" size="30" /><br/>
   <input type="submit" name="valider" value="Valider la saisie" />
   <input type="reset" name="effacer" value="Effacer le formulaire" />
  </form>
 <?php
 }
 ?>
 </body>
</html>
```

Figure 7.2 – Formulaire et traitement en un seul programme PHP.

Dans ce programme, la ligne d'en-tête du formulaire :

```
<form action="chaine_saisie_affichage_1page_web.php" method="post">
```

a été mise en commentaire HTML et remplacée par :

```
<form action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
```

Cette syntaxe utilise la variable super-globale `$_SERVER` (*cf.* chapitre 4).

- *Les problèmes de sécurité*

La première action d'un programme PHP récupérant les valeurs « texte » d'un entier ou d'un réel provenant d'un formulaire doit être explicitement de le convertir *via* `intval()` ou `floatval()` afin d'assurer la validité des informations.

Avec les chaînes de caractères, la donnée « texte » est directement fournie par le formulaire HTML, aucune conversion explicite de type n'est possible, ce qui peut occasionner des erreurs de traitements si la donnée est invalide.

Dans le programme précédent, si l'utilisateur saisit du texte avec des balises HTML, le programme PHP les récupère et les affiche telles quelles *via* `echo`. L'utilisateur peut alors *injecter* du code HTML *via* cette saisie non contrôlée, c'est un trou de sécurité nommé *injection HTML*. La figure 7.3 présente l'exécution du programme précédent avec la saisie de `<b>dupont</b>` dans le champ nom. L'affichage

du résultat montre que le texte `dupont` apparaît en gras car les balises `<b>` et `</b>` qui l'encadrent ont été affichées et interprétées par le navigateur.

Figure 7.3 – Injection HTML.

La saisie peut contenir des lignes JavaScript, et provoquer l'exécution de programmes non désirés sur votre ordinateur. Ainsi la saisie du texte `<script>alert('Piratage')</script>` fait apparaître une fenêtre d'alerte.

L'injection HTML est un trou de sécurité. Il faut traiter les données saisies pour empêcher toute injection de codes HTML dans les champs de saisie en : limitant la taille des champs de saisie et en utilisant des fonctions comme `htmlspecialchars()` ou `strip_tags()` pour neutraliser ou supprimer les balises HTML dans le texte saisi.

Transtypage

Pour forcer l'interprétation d'une variable en chaîne de caractères, il faut utiliser le préfixe `(string)`, comme le montrent les lignes de syntaxe suivantes :

```
$i=1234; // $i est un entier
$chaine=(string)$i; // $chaine est une chaîne de caractères
```

La syntaxe avec l'instruction `settype()` est de la forme :

```
settype($i,"string") ; // transtypage explicite
```

Les opérateurs

De chaînes

Le tableau 7.1 récapitule les opérateurs sur les chaînes de caractères.

Tableau 7.1 – Opérateur de chaînes de caractères

Opérateur	Signification	Exemple
.	Concaténation	`$NomComplet = $Nom . $Prenom;`
=	Affectation	`$Nom = "Dupont" ;`
.=	Concaténation et affectation	`$NomComplet = $Nom ;` `$NomComplet .= $Prenom ;`

De comparaison

Le tableau 7.2 récapitule les opérateurs de comparaison sur les chaînes.

Tableau 7.2 – Opérateur de comparaison sur les chaînes de caractères

Opérateur	Signification	Exemple
<	Inférieur à	`if ($ch1 < $ch2) …`
>	Supérieur à	`if ($ch1 > $ch2) …`
<=	Inférieur ou égal à	`if ($ch1 <= $ch2) …`
>=	Supérieur ou égal à	`if ($ch1 >= $ch2) …`
==	Est égal (après transtypage)	`if ($ch1 == $ch2) …`
!=	Différent (non égal) après transtypage	`if ($ch1 != $ch2) …`
<>	Différent (non égal) après transtypage	`if ($ch1 <> $ch2) …`

Nombre réel, notation française avec la virgule décimale

Problématique

Le format des réels utilise la notation anglo-saxonne avec le point décimal, et non la virgule. Cette section présente la saisie et l'affichage des nombres réels en français avec la virgule décimale, en utilisant la chaîne de caractères.

Environnement shell

Dans cet environnement il faut saisir une chaîne de caractères *via* la fonction `fscanf(STDIN,"%s",..)`, transformer les virgules en points décimaux *via* `str_replace()`, et convertir la chaîne en réel, *via* `floatval()`.

L'affichage utilise la fonction `setlocale()` qui a été présentée à la section 3.2 du chapitre 3. En voici quelques exemples de syntaxe :

```
setlocale(LC_NUMERIC,'fr_FR.utf8','fra');//Numériques fr+UTF8
setlocale(LC_ALL,'fr_FR.UTF-8');//Tous les paramètres fr+UTF8
setlocale(LC_ALL,'fr_FR'); //Tous les paramètres en français
setlocale(LC_ALL,'') ; //Paramètres selon environnement local
```

L'affichage au format monétaire avec une virgule et un espace pour les milliers, comme 1 543,97 €, utilise la fonction `number_format()`. La syntaxe est :

```
$Mont = number_format($Montant,2,","," ")." €";
```

Le programme `chaine_nombre_reel_shell.php` présente la saisie et l'affichage au format français. L'instruction `fscanf()` saisit une chaîne. Elle est convertie en réel *via* `str_replace()`, et `floatval()`. L'instruction `var_dump` confirme la conversion.

Trois affichages montrent : la donnée brute : **1234.45** ; la donnée au format français utilisant `setlocale()` : **1234,45** ; la donnée au format monétaire *via* `number_format()` et la concaténation du symbole € : **1 234,45 €**.

Listing 7.4 – Programme chaine_nombre_reel_shell.php

```
<?php
 echo "Montant: "            ;
 fscanf(STDIN,"%s",$Montant) ; // Saisie de la donnée
 // Remplace les virgules par des points décimaux
 $Montant = str_replace(",",".",$Montant) ;
 $Montant = floatval($Montant) ; // Conversion en réel
 // Affiche la donnée sans traitement
 echo "Montant saisi = $Montant".PHP_EOL;
 var_dump($Montant);
 // Dates, et numériques au format local (français)
 setlocale(LC_ALL, '') ;
 // Affiche la donnée réelle avec les paramètres nationaux
 echo "Montant au format français = $Montant".PHP_EOL;
 // Affiche la donnée réelle en format monétaire français
 $Mont = number_format($Montant,2,","," ")." €"            ;
 echo "Montant au format monétaire français = $Mont".PHP_EOL;
?>
```

Voici son exécution :

Listing 7.5 – Exécution de chaine_nombre_reel_shell.php

```
$ php chaine_nombre_reel_shell.php
Montant: 1234,45
Montant saisi = 1234.45
float(1234.45)
Montant au format français = 1234,45
Montant au format monétaire français = 1 234,45 €
```

Environnement web

La saisie par formulaire est abordée à la section 3.3 du chapitre 3.

Le fichier `chaine_nombre_reel_web.html` présente le formulaire. La variable Montant de type « text » est transmise *via* la méthode POST au programme PHP.

Listing 7.6 – Fichier chaine_nombre_reel_web.html

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
  <meta charset="utf-8" />
  <title>Saisie R&eacute;el en Fran&ccedil;ais</title>
 </head>
 <body>
  <form action="chaine_nombre_reel_web.php" method="post">
```

```
    <fieldset>
    <legend>Saisissez un nombre r&eacute;el au format fran&ccedil;ais
    :</legend><br/>
     Montant (exemple : 1234,45): <input type="text" name="Montant"
     size="8" /> &euro;<br/><br/>
     <input type="submit" value="Valider" />
     <input type="reset" value="Effacer le formulaire" />
    </fieldset>
   </form>
  </body>
</html>
```

Le programme `chaine_nombre_reel_web.php` traite et affiche le nombre.

Listing 7.7 – Programme chaine_nombre_reel_web.php

```
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Saisie R&eacute;el en Fran&ccedil;ais</title>
 </head>
 <body>
  <?php
  define("WEB_EOL","<br />");
  // Récupération des données
  $Montant     = $_POST['Montant'] ;
  // Remplace les virgules par des points décimaux
  $Montant     = str_replace(",",".",$Montant) ;
  // Conversion dans le type réel
  $Montant     = floatval($Montant) ;
  // Affiche la donnée réelle sans traitement
  echo "Montant saisi = $Montant".WEB_EOL ;
  var_dump($Montant)                    ;
  echo WEB_EOL ;
  // Dates, valeurs numériques au format local (français)
  setlocale (LC_ALL, 'fr_FR');
  // Affiche la donnée réelle avec les paramètres nationaux
  echo "Montant au format fran&ccedil;ais = $Montant".WEB_EOL;
  // Affiche la donnée réelle en format monétaire français
  $Mont = number_format($Montant,2,","," ")." €" ;
  echo "Montant au format mon&eacute;taire fran&ccedil;ais = $Mont".
  WEB_EOL;
  ?>
 </body>
</html>
```

La figure 7.4 présente l'écran de saisie et la page après validation.

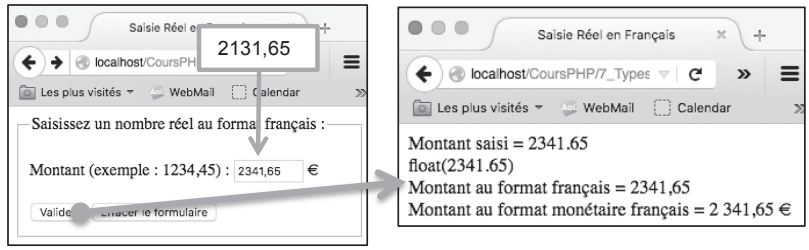


Figure 7.4 – Saisie et affichage d'un réel au format français.

Les fonctions

Le tableau 7.3 présente les fonctions sur les chaînes de caractères. La liste complète est disponible à l'URL : *http://php.net/manual/fr/ref.strings.php*.

Tableau 7.3 – Fonctions sur les chaînes de caractères

Fonction	Signification	Exemple
`addslashes`	Ajoute des antislashs dans une chaîne	`$ch="Aujourd'hui";` `$res=addslashes($ch);`
`crypt`	Hachage à sens unique (indéchiffrable)	`$mdp=crypt('MotdePasse');`

`ctype_alnum`
`ctype_alpha`
`ctype_digit`
	Vérifie qu'une chaîne est alphanumérique, alphabétique, un entier	`if (ctype_alnum($ch)) …`
		`if (ctype_alpha($ch)) …`
		`if (ctype_digit($ch)) …`

`ctype_lower` `ctype_upper`	Vérifie qu'une chaîne est en minuscules/majuscules	`if (ctype_lower($ch)) …` `if (ctype_upper($ch)) …`
`explode`	Coupe une chaîne en segments	`$ch="au;revoir;salut";` `$tab=explode(";",$ch);`
`htmlspecialchars`	Convertit les caractères spéciaux en HTML	`$N=htmlspecialchars($N);`
`iconv()`	Convertit une chaîne dans un jeu de caractères	`$res=iconv("UTF-8","I-SO-8859-1//IGNORE",$texte) ;`
`implode`	Rassemble les cases d'un tableau en une chaîne	`$tab=array('meri','jean')` `$ch=implode(";", $tab);`
`money_format`	Met un nombre au format monétaire	`$val=1234.5672;` `$ch=money_format('%#5.2n',$val);`

`number_format`	Formate un nombre pour l'affichage	`$val=1234.5672;`
		`$ch=number_format($val,2,",","`
		`")."&euro;";`

`preg_replace`	Recherche et remplace par expression rationnelle standard	`$saisie= preg_`
		`replace('/\s{2,}/',' ', $saisie);`
		`//remplace les espaces multiples par un seul`

Fonction	Signification	Exemple
setlocale	Modifie la localisation	`setlocale(LC_ALL,'fr_FR');`
sha1	Calcule le sha1 d'une chaîne de caractères	`$mdp=sha1('MotdePasse');`
sprintf	Retourne une chaîne formatée	`$ch=sprintf("res:%s",$i);`

sscanf	Analyse une chaîne à l'aide d'un format	`$date="03/02/2015";`
		`sscanf($date,"%d/%d/%d"`
		`,$j,$m,$a);`

str_replace	Remplace toutes les occurrences dans une chaîne	`$voy=array("a","e","i","o"`
		`,"u","A","E","I","O","U");`
		`$res=str_replace($voy,"", "Bon-`
		`jour");`

str_split	Convertit une chaîne en tableau	`$ch="Jean Claude";` `$tab=str_split($str,3);`
strcmp	Comparaison binaire	`if (strcmp($ch1,$ch2)!=0)`
strip_tags	Supprime les balises HTML et PHP	`$nom=strip_tags($nom);`

stripslashes	Supprime les antislashs d'une chaîne	`//$res="Aujourd\'hui";`
		`$ch=stripslashes($res);`
		`//$ch= "Aujourd'hui"`

strlen	Retourne la taille	`$taille=strlen($ch1);`

strpos	Cherche la position de la première occurrence dans une chaîne	`$chaine='ABCDEFabcdef';`
		`$ch1='Fa';`
		`pos=strpos($chaine,$ch1);`

strstr	Trouve la première occurrence dans une chaîne	`$addr='Jean@cnam.fr';` `$dom=strstr($addr,'@');`

strtolower
strtoupper
	Renvoie une chaîne en minuscules/majuscules	`$ch='BONJOUR';`
		`$chlower=strtolower($ch);`
		`$chlupper=strtoupper($ch);`

strtr	Remplace des caractères dans une chaîne	`$ch=strtr($ch,"a","A");`
substr	Retourne un segment de chaîne	`$reste=substr($ch,3,2);`
trim	Supprime les espaces (ou fin de ligne) au début et fin	`$trimch1 = trim($ch1);`
ucwords	Met en majuscule la première lettre des mots	`$chu=ucwords($ch);`
utf8_encode utf8_decode	Convertit de ISO-8859-1 en UTF-8 et inverse	`$res=utf8_encode($ch) ;` `$res=utf8_decode($ch) ;`

Remarque

Des fonctions préfixées « iconv_ » effectuent la conversion des chaînes de caractères dans un autre jeu de caractère, en particulier la fonction `iconv()`.

Exemple de programme

Le programme `chaine_exemple_shell.php` présente un traitement de chaîne.

Listing 7.8 – Programme chaine_exemple_shell.php

```
<?php
 echo "Entrez Nom : "         ;
 fscanf(STDIN,"%s",$Nom)      ;
 echo "Entrez Prénom : "      ;
 fscanf(STDIN,"%s",$Prenom)   ;
 $NomComplet=$Nom." ".$Prenom ;
 $InitialeNom= $Nom[0]        ;
 $longueur=strlen($Nom)       ;
 echo "Le Nom complet est     : ".$NomComplet.PHP_EOL ;
 echo "La longueur du nom est : ".$longueur.PHP_EOL   ;
 echo "L'initiale  est        : ".$InitialeNom.PHP_EOL;
?>
```

Voici son exécution :

Listing 7.9 – Exécution de chaine_exemple_shell.php

```
Entrez Nom : Dupont
Entrez Prénom : Jean
Le Nom complet est     : Dupont Jean
La longueur du nom est : 6
L'initiale  est        : D
```

7.3 Les tableaux

Définition

Un tableau est une variable contenant *un ensemble de cases*, chacune pouvant être du même type ou de type différent. On peut définir un tableau d'entiers, de réels, ou de caractères, mais aussi un tableau de tableaux, un tableau de chaînes de caractères, un tableau d'objets ou un tableau mixant différents types. Un tableau peut être à une, deux, trois, *N* dimensions.

La figure 7.5 présente trois tableaux :

- `$tab_ent` de 1 000 cases numérotées de 0 à 999, contenant des entiers ;
- `$tab_noms` de 100 cases numérotées de 0 à 99, contenant des chaînes ;
- `$image` à deux dimensions de 500 lignes numérotées de 0 à 499 et 1 000 colonnes numérotées de 0 à 999, contenant des entiers.

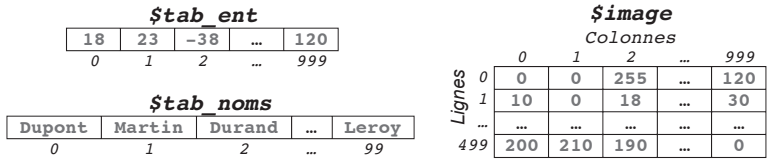


Figure 7.5 – Structure de différents tableaux.

Caractéristiques des tableaux en PHP

Tableaux dynamiques

Les tableaux en PHP sont **dynamiques**, c'est-à-dire qu'ils « grandissent » automatiquement au fur et à mesure qu'on leur affecte des données. Le nombre de cases est identique au nombre d'éléments qu'ils contiennent.

Tableaux numérotés, associatifs et mixtes

Le langage PHP propose deux types de tableaux :

- les tableaux *numérotés* : les numéros des cases, les indices, sont des entiers ;
- les tableaux *associatifs* : les cases sont étiquetées avec des noms.

Pour chaque type, il est possible d'avoir des cases dont la nature du contenu varie. Les tableaux numérotés ou associatifs sont alors *mixtes*.

La procédure d'affichage print_r

Quelle que soit la dimension ou le type du tableau, le langage PHP propose une procédure d'affichage de sa structure et de son contenu : `print_r()`. Sa syntaxe est :

```
print_r($tab_ent)  ; // Affichage du tableau $tab_ent
```

Les tableaux numérotés

Principe

Chaque case du tableau est numérotée avec une valeur numérique allant par exemple de 0 à 9 pour un tableau de 10 cases.

Déclaration

La déclaration d'un tableau utilise la fonction `array()`, mais il est possible d'affecter les valeurs à la création du tableau, ou après, case par case.

- *Tableau à une dimension*

Numérotation implicite des indices

Le programme `tableau_creation1_shell.php` déclare trois tableaux à une dimension : `$tab_ent`, `$tab_notes`, `$tab_noms`, qui sont des tableaux d'entiers, de réels et de chaînes de caractères. Les données y sont rangées lors de la création :

- `$tab_ent` contient les valeurs : 18, –25 et 88 ;
- `$tab_notes` contient les valeurs : 0.5, 15.5, 8.0, 20.0 ;
- `$tab_noms` contient les valeurs : 'Dupont', 'Martin', 'Durand' ;

Listing 7.10 – Programme tableau_creation1_shell.php

```
<?php
// Création des tableaux
$tab_ent   = array(18,-25,88);                  //d'entiers
$tab_notes = array(0.5,15.5,8.0,20.0);          //de réels
$tab_noms  = array('Dupont','Martin','Durand');//de chaînes
echo '--- Affichage de $tab_ent ---'.PHP_EOL;
print_r($tab_ent)  ;
echo '--- Affichage de $tab_notes ---'.PHP_EOL;
print_r($tab_notes);
echo '--- Affichage de $tab_noms ---'.PHP_EOL;
print_r($tab_noms) ;
?>
```

Lors de son exécution, `print_r()` affiche la structure et le contenu des tableaux.

Listing 7.11 – Exécution de tableau_creation1_shell.php

```
--- Affichage de $tab_ent ---
Array
(
    [0] => 18
    [1] => -25
    [2] => 88
)
--- Affichage de $tab_notes ---
Array
(
    [0] => 0.5
    [1] => 15.5
    [2] => 8
    [3] => 20
)
```

```
--- Affichage de $tab_noms ---
Array
(
    [0] => Dupont
    [1] => Martin
    [2] => Durand
)
```

Remarques

Les cases sont automatiquement numérotées à partir de l'indice 0.

Il est possible d'affecter manuellement le tableau et de laisser PHP numéroter les cases. Voici les syntaxes qui créent manuellement le tableau `$tab_ent`.

```
// Création d'un tableau d'entiers
$tab_ent[] =  18;
$tab_ent[] = -25;
$tab_ent[] = -88;
```

Numérotation explicite des indices

L'indice peut être indiqué explicitement. Les indices sont une suite continue de valeurs, cas de `$tab_noms`, ou une suite discontinue comme avec `$tab_ent`.

```
// Création d'un tableau d'entiers. Indices discontinus
$tab_ent[-1]=18;
$tab_ent[10]=-25;
$tab_ent[30]=-88;
// Création d'un tableau de chaînes. Indices continus
$tab_noms[0]='Dupont';
$tab_noms[1]='Martin';
$tab_noms[2]='Durand';
```

La dernière syntaxe présente la création du tableau numéroté `$tab_ent` *via* la fonction `array()`, avec des indices discontinus.

```
// Création d'un tableau d'entiers
$tab_ent = array (
   -1 =>  18,
   10 => -25,
   30 =>  88);
```

- *Tableau à deux dimensions*

Numérotation implicite des indices

Le programme `tableau_creation6_shell.php` présente trois tableaux à deux dimensions : `$tab2_ent`, `$tab2_notes`, `$tab2_noms`, qui sont des tableaux d'entiers, de réels et de chaînes de caractères.

Listing 7.12 – Programme tableau_creation6_shell.php

```
<?php
// Création d'un tableau d'entiers
$tab2_ent=array(array(18,22),array(-25,-3),array(-5,88,120));
// Création d'un tableau de réels
$tab2_notes=array (array(0.5,15.5),array(8.0,20.0));
// Création d'un tableau de chaînes de caractères
$tab2_noms=array(array('Dupont', 'Martin'),
                 array('Durand','Mery'));
echo '--- Affichage de $tab2_ent ---'.PHP_EOL;
print_r($tab2_ent)  ;
echo '--- Affichage de $tab2_notes ---'.PHP_EOL;
print_r($tab2_notes);
echo '--- Affichage de $tab2_noms ---'.PHP_EOL;
print_r($tab2_noms) ;
?>
```

La figure 7.6 présente ces trois tableaux. La particularité de `$tab2_ent` est que seule la dernière ligne possède trois colonnes.

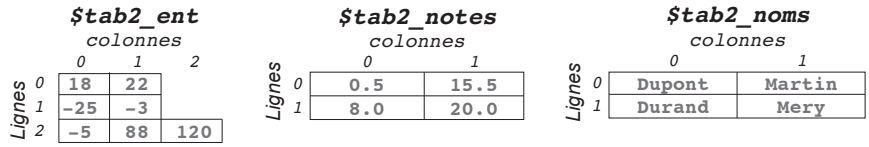


Figure 7.6 – Tableaux à deux dimensions.

Remarque

Dans un tableau à N dimensions, chaque ligne peut avoir un nombre différent de colonnes.

`print_r()` affiche la structure des tableaux (seul le premier est présenté).

Listing 7.13 – Exécution de tableau_creation6_shell.php

```
$ php tableau_creation6_shell.php
--- Affichage de $tab2_ent ---
Array
(
    [0] => Array
        (
            [0] => 18
            [1] => 22
        )
    [1] => Array
        (
            [0] => -25
            [1] => -3
        )
    [2] => Array
```

```
        (
            [0] => -5
            [1] => 88
            [2] => 120
        )
)
...
```

Numérotation explicite des indices

Il est également possible de préciser l'indice explicitement. Les lignes suivantes créent un tableau et rangent les valeurs dans les cases dont l'indice des lignes est précisé. L'indice des colonnes n'est pas systématiquement indiqué en particulier pour les lignes 0 et 2 du tableau d'entiers `$tab2_ent`.

```
// Création d'un tableau d'entiers
$tab2_ent[0][]=18  ; // Première ligne
$tab2_ent[0][]=22  ;
$tab2_ent[1][0]=-25; // Deuxième ligne
$tab2_ent[1][1]=-3 ;
$tab2_ent[2][]=-5  ; // Troisième ligne
$tab2_ent[2][]=88  ;
$tab2_ent[2][]=120 ;
```

Il est également possible de donner une suite discontinue d'indices comme cela est présenté dans les syntaxes suivantes.

```
// Création d'un tableau d'entiers
$tab2_ent[-1][1]=18; //Première ligne, numérotée : -1
$tab2_ent[-1][9]=22;
$tab2_ent[1][3]=-25; //Deuxième ligne, numérotée :  1
$tab2_ent[1][-1]=-3;
$tab2_ent[5][]=-5  ; //Troisième ligne, numérotée:  5
$tab2_ent[5][]=88  ;
$tab2_ent[5][]=120 ;
// Création d'un tableau de chaînes de caractères
$tab2_noms[-1][2]='Dupont'; //Première ligne, numérotée : -1
$tab2_noms[-1][0]='Martin';
$tab2_noms[10][3]='Durand'; //Deuxième ligne, numérotée : 10
$tab2_noms[10][6]='Mery'  ;
```

La figure 7.7 présente ces deux tableaux.

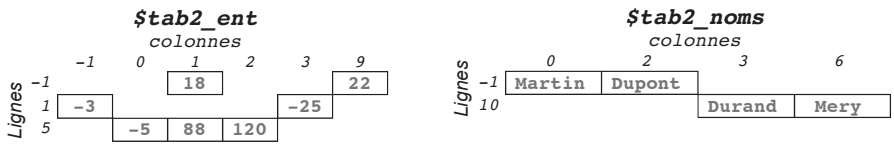


Figure 7.7 – Tableaux à deux dimensions avec indices discontinus.

La dernière syntaxe crée le tableau `$tab2_ent` précédent *via* `array()`.

```
$tab2_ent = array (
   -1 => array(1 =>  18, 9 => 22),
    1 => array(3 => -25,-1 => -3),
    5 => array(-5,88,120));
```

Les tableaux associatifs

Principe

Les tableaux associatifs fonctionnent sur le même principe que les tableaux numérotés, mais les indices numérotés sont remplacés par des *étiquettes nommées*.

Ce type de tableau permet, par exemple, de mémoriser le code postal d'une ville dont le nom devient l'étiquette de la case. Le tableau `$CP` suivant possède une case ayant l'étiquette `'CRETEIL'` dont le contenu est l'entier `94000`.

```
$CP['CRETEIL']=94000 ;
```

Déclaration

La déclaration d'un tableau associatif utilise la fonction `array()`, mais il est possible d'affecter les valeurs dès la création du tableau, ou après sa création, case par case. Nous présentons les syntaxes pour des tableaux à une et à deux dimensions.

- *Tableau à une dimension*

Le programme `tableau_a_creation1_shell.php` présente la déclaration de deux tableaux associatifs `$CP` et `$DEPT`, qui sont respectivement des tableaux d'entiers et de chaînes de caractères. Le premier contient les codes postaux des villes et le second le nom de leur département d'appartenance.

Listing 7.14 – Programme tableau_a_creation1_shell.php

```
<?php
$CP = array (// Tableau des codes postaux
    'CRETEIL'    => 94000,
    'NICE'       => 6000,
    'LA BAULE'   => 44500,
    'STRASBOURG' => 67000);
$DEPT = array (// Tableau des départements d'appartenance
    'CRETEIL'    => 'Val-de-Marne',
    'NICE'       => 'Alpes-Maritimes',
    'LA BAULE'   => 'Loire-Atlantique',
    'STRASBOURG' => 'Bas-Rhin');
echo '--- Affichage de $CP ---'.PHP_EOL;
print_r($CP)  ;
echo '--- Affichage de $DEPT ---'.PHP_EOL;
print_r($DEPT) ;
?>
```

La figure 7.8 présente ces deux tableaux.

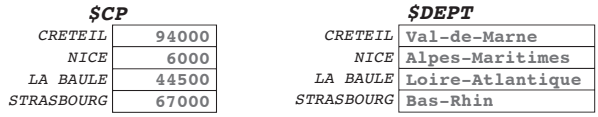


Figure 7.8 – Tableaux associatifs.

La fonction `print_r()` affiche la structure et le contenu de chaque tableau.

Listing 7.15 – Exécution de tableau_a_creation1_shell.php

```
$ php tableau_a_creation1_shell.php
--- Affichage de $CP ---
Array
(
    [CRETEIL] => 94000
    [NICE] => 6000
    [LA BAULE] => 44500
    [STRASBOURG] => 67000
)
--- Affichage de $DEPT ---
Array
(
    [CRETEIL] => Val-de-Marne
    [NICE] => Alpes-Maritimes
    [LA BAULE] => Loire-Atlantique
    [STRASBOURG] => Bas-Rhin
)
```

Les codes postaux démontrent qu'une analyse des données est toujours indispensable. En effet, certains codes postaux commencent par 0 comme Nice dont le code postal est 06000. Ce code doit être noté 6000 et non 06000 pour éviter son interprétation en octal (commençant par 0), sinon le code postal de Nice serait 3072 en base 10 soit 06000 en octal ! Enfin, Les codes postaux sont présentés comme des numériques, mais il est préférable de les traiter en chaînes de caractères pour régler le cas de la Corse dont les départements sont 2A et 2B.

Les syntaxes suivantes créent manuellement `$CP` avec deux nouveaux éléments.

```
$CP['CRETEIL']    = 94000 ; // Tableau des codes postaux
$CP['NICE']       = 6000  ;
$CP['LA BAULE']   = 44500 ;
$CP['STRASBOURG'] = 67000 ;
$CP['creteil']    = 94001 ; // Sensible à la casse
$CP['créteil']    = 94002 ;
```

Remarque

Les étiquettes des tableaux associatifs sont sensibles à la casse. Ainsi la case 'CRETEIL' est différente de la case 'creteil' ou 'créteil'. D'autre part il faut éviter les étiquettes

contenant des accents du fait de la non-universalité des tables de codage des caractères accentués.

- *Tableau à deux dimensions*

Le programme `tableau_a_creation3_shell.php` suivant présente la déclaration d'un tableau à deux dimensions : `$CP_DEPT`.

Listing 7.16 – Programme tableau_a_creation3_shell.php

```
<?php
// Tableau des codes postaux
$CP_DEPT = array (
  'CRETEIL'=> array('CP'=>94000,'DEPT'=>'Val-de-Marne'),
  'NICE'=> array('CP'=>6000,'DEPT'=>'Alpes-Maritimes'),
  'LA BAULE'=> array('CP'=>44500,'DEPT'=>'Loire-Atlantique'),
  'STRASBOURG'=> array('CP'=>67000,'DEPT'=>'Bas-Rhin'));
// affichage du tableau
echo '--- Affichage de $CP_DEPT ---'.PHP_EOL;
print_r($CP_DEPT)  ;
?>
```

La figure 7.9 présente ce tableau.

$CP_DEPT

colonnes

Lignes	*CP*	*DEPT*
CRETEIL	94000	Val-de-Marne
NICE	6000	Alpes-Maritimes
LA BAULE	44500	Loire-Atlantique
STRASBOURG	67000	Bas-Rhin

Figure 7.9 – Tableaux associatifs à deux dimensions.

Voici l'exécution de ce programme :

Listing 7.17 – Exécution de tableau_a_creation3_shell.php

```
$ php tableau_a_creation3_shell.php
--- Affichage de $CP_DEPT ---
Array
(
    [CRETEIL] => Array
        (
            [CP] => 94000
            [DEPT] => Val-de-Marne
        )
    [NICE] => Array
        (
            [CP] => 6000
            [DEPT] => Alpes-Maritimes
        )
    [LA BAULE] => Array
        (
```

```
            [CP] => 44500
            [DEPT] => Loire-Atlantique
        )
    [STRASBOURG] => Array
        (
            [CP] => 67000
            [DEPT] => Bas-Rhin
        )
)
```

Les tableaux mixtes

Principe

Les tableaux mixtes ont la particularité de posséder des cases pouvant être de type différent. Cette organisation mémorise des structures de données généralement représentées sous la forme « d'enregistrements » dans d'autres langages.

Déclaration

La déclaration de ce type de tableau est identique à ce qui a été présenté. Elle est effectuée *via* la fonction `array()`, ou par l'affectation de valeurs.

Exemple

Les syntaxes ci-dessous présentent la déclaration de trois tableaux mixtes :

- `$tabm_references` est un tableau *numéroté* à une dimension ;
- `$tabm_produits` est un tableau *associatif* à une dimension ;
- `$tabm_stock` est un tableau *numéroté* et *associatif* à trois dimensions.

```
// Création d'un tableau mixte numéroté à une dimension
$tabm_reference = array (
    0      => 10000,
    1      => 'articles de sports',
    2012   => 8358.34,
    2013   => 10329.22,
    2014   => 12654.67);
// Création d'un tableau mixte associatif à une dimension
$tabm_produit = array (
    'ref'           => 1002,
    'libelle'       => 'Chaussures de tennis',
    'quantite'      => 30,
    'prix_unitaire' => 95.99);
// Création d'un tableau mixte à trois dimensions
$tabm_stock =
 array (
   1002 => array(
     'libelle'  => 'Chaussures de tennis',
```

```
    'quantite' => 30,
    'prix'     => 95.99,
    'CA'  => array(2012=>1247.87,2013=>1151.88,2014=>1631.83)
     ),
  1003 => array(
    'libelle'  => 'Raquettes de tennis',
    'quantite' => 200,
    'prix'     => 100.65,
    'CA'  => array(2012=>805.20,2013=>905.85,2014=>1006.50)
     )
);
```

La figure 7.10 présente ces tableaux. `$tabm_produit` contient des éléments s'apparentant aux *champs d'une donnée*. On y trouve la référence, le libellé, la quantité et le prix unitaire d'un produit. La conversion de ce tableau en un tableau à deux dimensions permet de gérer un stock d'un magasin. `$tabm_stock` présente ce tableau, avec le chiffre d'affaires de chaque produit selon l'année.

$tabm_reference

0	10000
1	'articles de sports'
2012	8358.34
2013	10329.22
2014	12654.67

$tabm_produit

ref	1002
libelle	'Chaussures de tennis'
quantite	30
prix_unitaire	95.99

$tabm_stock

Lignes	libelle	quantite	prix	CA		
				2012	2013	2014
1002	Chaussures de tennis	30	95.99	1247.87	1151.88	1631.83
1003	Raquettes de tennis	200	100.65	805.20	905.85	1006.50

Figure 7.10 – **Tableaux mixtes à une ou N dimensions.**

Les listes

Principe

L'instruction `list()`rassemble des variables indépendantes sous la forme d'un tableau, pour les assigner en une seule ligne. La syntaxe générale est :

```
list($var1,$var2,$var3) = $tab;
```

où `$var1`, `$var2` et `$var3` sont trois variables et `$tab` le tableau les affectant. Cette syntaxe est identique à (dans cet ordre) :

```
$var3 = $tab[2];
$var2 = $tab[1];
$var1 = $tab[0];
```

Il est possible d'omettre un élément afin de ne récupérer que certains éléments. Cet exemple affecte les variables `$var1` et `$var2` à partir des cases 0 et 2 de `$tab`.

```
list($var1,, $var2) = $tab;
```

`list()` permet également d'affecter des cases d'un tableau dans un autre. Cet exemple affecte les cases 0, 1 et 2 de `$t` avec les cases 0, 1 et 2 de `$tab`.

```
list($t[0], $t[1], $t[2]) = $tab;
```

Les indices du tableau précédant `$t` peuvent être des étiquettes, ce qui permet de « convertir » un tableau numéroté en tableau associatif :

```
list($ta['Nom'], $ta['Prenom'], $ta['Age']) = $tab;
```

Enfin, on peut imbriquer l'instruction `list()` pour affecter des variables à partir d'un tableau à deux dimensions.

```
list($var1, list($var2, $var3)) = $tab;
```

Dans cet exemple, le tableau `$tab` contient une case 0, et une case 1 qui est séparée en deux colonnes 0 et 1. Le contenu de la case 0 de `$tab` affectera la variable `$var1`. Le contenu de la colonne 0 de la case 1 affectera la variable `$var2`, et le contenu de la colonne 1 de la case 1 affectera la variable `$var3`.

Remarque

L'instruction list() assigne les valeurs en commençant par la valeur la plus en droite. Dans le cas de variables indépendantes, cela n'a aucune incidence ; dans le cas de tableaux, ce sera l'indice le plus à droite qui sera affecté le premier. list() ne fonctionne que sur des tableaux à index numériques avec une indexation commençant à 0.

Exemple

Le programme `tableau_liste1_shell.php` présente l'affectation des variables `$Nom`, `$Prenom` et `$Age` à partir du tableau `$une_personne` *via* `list()`.

Listing 7.18 – Programme tableau_liste1_shell.php

```
<?php
// Création d'un tableau
$une_personne = array ('Martin','Pierre',25);
// Affectation de trois variables distinctes
list($Nom, $Prenom, $Age) = $une_personne;
// Affichage
echo "Nom    = $Nom".PHP_EOL;
echo "Prénom = $Prenom".PHP_EOL;
echo "Age    = $Age".PHP_EOL;
?>
```

Voici son exécution :

Listing 7.19 – Exécution de tableau_liste1_shell.php

```
$ php tableau_liste1_shell.php
Nom    = Martin
Prénom = Pierre
Age    = 25
```

Le traitement des tableaux

Cette section présente le traitement des tableaux, en particulier la saisie des données, la suppression d'éléments, le parcours et la recherche.

Afin de nous concentrer sur le traitement des tableaux indépendamment du contexte, les exemples proposés sont en environnement shell.

La saisie des éléments d'un tableau

- *Tableau à une dimension*

Numéroté

Le programme `tableau_saisie1_shell.php` montre comment affecter les données saisies dans un tableau à une dimension dans le cas d'une liste d'entiers (`$tab_ent`), de chaînes de caractères (`$tab_noms`) et de données mixtes (`$une_personne`). Pour chaque type de saisie, on utilise :

- la fonction `fgets()` qui lit la saisie complète dans un environnement shell ;
- la fonction `trim()` qui supprime les espaces, au début et en fin de chaîne, ainsi que le saut de ligne en fin de chaîne ;
- la fonction `preg_replace()` qui remplace une suite d'espaces par un seul ; la syntaxe est : `$saisie=preg_replace('/\s{2,}/',' ',$saisie)`, où le premier argument est le motif de recherche, le deuxième est la chaîne de remplacement, le troisième est la chaîne sur laquelle porte le traitement. Le motif de recherche contient « `\s` » qui représente un espace, ou une tabulation, et « `{2,}` » indique le nombre d'occurrences, ici au moins deux occurrences ;
- la fonction `explode()` qui transforme une chaîne de caractères, en un tableau. Le délimiteur des éléments de la chaîne est indiqué en premier argument.

Le traitement de la liste de noms et de données mixtes utilise un traitement supplémentaire, car les noms peuvent contenir des espaces. Le caractère délimiteur utilisé est « `;` » pour la fonction `explode()`. Il faut supprimer les espaces avant et après le « `;` » afin que les noms saisis soient sans espace ni au début ni à la fin.

Listing 7.20 – Programme tableau_saisie1_shell.php

```
<?php
// --- Saisie d'une liste d'entiers ---
echo "Entrez une suite d'entiers (ex : 10  -3  8) : ";
$saisie=fgets(STDIN);
// --- Traitement de la chaîne lue, ---
// Suppression des espaces (début,fin) et du saut de ligne
$saisie=trim($saisie);
// Remplace les espaces multiples par un seul espace
$saisie= preg_replace('/\s{2,}/', ' ', $saisie);
$tab_ent=explode(' ',$saisie);
```

```
echo '--- Affichage de $tab_ent ---'.PHP_EOL;
print_r($tab_ent)  ;
// --- Saisie d'une liste de noms ---
echo "Entrez une suite de noms (ex:Dupont;Martin;Durand) : ";
$saisie=fgets(STDIN);
// --- Traitement de la chaîne lue, ---
// Suppression des espaces (début,fin) et du saut de ligne
$saisie=trim($saisie);
// Remplace les espaces multiples par un seul espace
$saisie= preg_replace('/\s{2,}/',' ',$saisie);
// Remplace ; suivi d'un espace par ;
$saisie= preg_replace('/;\s/',';',$saisie);
// Remplace un espace suivi d'un ; par ;
$saisie= preg_replace('/\s;/',';',$saisie);
// Rangement dans le tableau
$tab_noms=explode(';',$saisie);
echo '--- Affichage de $tab_noms ---'.PHP_EOL;
print_r($tab_noms) ;
// --- Saisie d'une liste de données mixtes ---
echo "Entrez un nom,un prénom et un âge (ex:Dupont;Jean;28):";
$saisie=fgets(STDIN);
// --- Traitement de la chaîne lue, ---
// Suppression des espaces (début,fin) et du saut de ligne
$saisie=trim($saisie);
// Remplace les espaces multiples par un seul espace
$saisie= preg_replace('/\s{2,}/',' ',$saisie);
// Remplace ; suivi d'un espace par ;
$saisie= preg_replace('/;\s/',';',$saisie);
// Remplace un espace suivi d'un ; par ;
$saisie= preg_replace('/\s;/',';',$saisie);
$une_personne=explode(';',$saisie);//Rangement dans le tableau
echo '--- Affichage de $une_personne ---'.PHP_EOL;
print_r($une_personne) ;
?>
```

Voici son exécution :

Listing 7.21 – Exécution de tableau_saisie1_shell.php

```
$ php tableau_saisie1_shell.php
Entrez une suite d'entiers (ex : 10  -3  8) : -5  7  18
--- Affichage de $tab_ent ---
Array
(
    [0] => -5
    [1] => 7
    [2] => 18
)
Entrez une suite de noms (ex : Dupont;Martin;Durand) : De  la
Fontaine ; Martin  ; Durand
--- Affichage de $tab_noms ---
```

```
Array
(
    [0] => De la Fontaine
    [1] => Martin
    [2] => Durand
)
Entrez un nom, un prénom et un âge (ex : Dupont;Jean;28) : De  la
Marre ; Jean Philippe ; 32
--- Affichage de $une_personne ---
Array
(
    [0] => De la Marre
    [1] => Jean Philippe
    [2] => 32
)
```

Remarques

Les tableaux présentés ont des indices numérotés. Pour le tableau `$une_personne`, il serait préférable d'avoir un tableau associatif avec des étiquettes `'Nom'`, `'Prenom'` et `'Age'`.

Associatif

Afin d'obtenir un tableau associatif avec les étiquettes `'Nom'`, `'Prenom'` et `'Age'`, pour le tableau mixte précédent `$une_personne`, on utilise la syntaxe `list()`. La syntaxe de rangement dans le tableau du programme précédent devient :

```
list($une_personne['Nom'],$une_personne['Prenom'],$une_personne['Age'])
= explode(';',$saisie);
```

- *Tableau à deux dimensions*

Numéroté

Le programme `tableau_saisie2_shell.php` montre la saisie d'une liste de personnes et son rangement dans un tableau numéroté à deux dimensions.

La saisie s'arrête quand la touche ENTREE est validée sans aucune donnée. Chaque ligne contient les informations d'une personne. Les colonnes 0, 1 et 2 correspondent respectivement au nom, prénom et âge. Une boucle `while` continue tant que la saisie n'est pas vide (`!empty($saisie)`).

Listing 7.22 – Programme tableau_saisie2_shell.php

```
<?php
// --- Saisie d'une liste de personnes ---
$saisie="saisie non vide";
while (!empty($saisie))
{echo "Entrez un nom,un prénom et un âge(ex:Dupont;Jean;28):";
 $saisie=fgets(STDIN);
 // --- Traitement de la chaîne lue, ---
 // Suppression des espaces (début,fin) et du saut de ligne
```

```
 $saisie=trim($saisie);
 // Remplace les espaces multiples par un seul espace
 $saisie= preg_replace('/\s{2,}/',' ',$saisie);
 // Remplace ; suivi d'un espace par ;
 $saisie= preg_replace('/;\s/',';',$saisie);
 // Remplace un espace suivi d'un ; par ;
 $saisie= preg_replace('/\s;/',';',$saisie);
 // On vérifie que la saisie n'est pas vide
 if (!empty($saisie)) // Rangement dans le tableau
 { $une_personne=explode(';',$saisie);
   $tab_personnes[]=$une_personne    ;
 }
}
echo '--- Affichage de $tab_personnes ---'.PHP_EOL;
print_r($tab_personnes) ;
?>
```

Voici son exécution :

Listing 7.23 – Exécution de tableau_saisie2_shell.php

```
$ php tableau_saisie2_shell.php
Entrez un nom, un prénom et un âge (ex : Dupont;Jean;28) :
Dupont;Jean;28
Entrez un nom, un prénom et un âge (ex : Dupont;Jean;28) : De  la
Marre ; Jean Philippe ; 32
Entrez un nom, un prénom et un âge (ex : Dupont;Jean;28) : Martin ;
Pierre;25
Entrez un nom, un prénom et un âge (ex : Dupont;Jean;28) :
--- Affichage de $tab_personnes ---
Array
(
    [0] => Array
        (
            [0] => Dupont
            [1] => Jean
            [2] => 28
        )
    [1] => Array
        (
            [0] => De la Marre
            [1] => Jean Philippe
            [2] => 32
        )
    [2] => Array
        (
            [0] => Martin
            [1] => Pierre
            [2] => 25
        )
)
```

La figure 7.11 présente le tableau `$tab_personnes` ainsi construit.

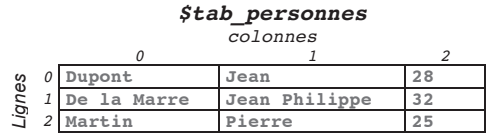


Figure 7.11 – Tableaux numérotés de personnes.

Associatif

Le programme précédent construit un tableau numéroté. Si la numérotation des lignes est logique car chaque ligne contient une personne, les colonnes représentant les « champs » différents d'une personne devraient être nommées. Voici les lignes à modifier dans le programme précédent pour obtenir un tableau associatif avec les étiquettes `'Nom'`, `'Prenom'`, `'Age'` pour les colonnes, grâce à la syntaxe `list()` :

```
if (!empty($saisie)) // Rangement dans le tableau associatif
{
list($une_personne['Nom'],$une_personne['Prenom'],$une_
personne['Age'])=explode(';',$saisie);
  $tab_personnes[]=$une_personne    ;
}
```

La figure 7.12 présente ce nouveau tableau `$tab_personnes`.

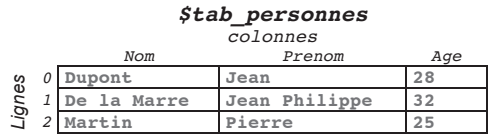


Figure 7.12 – Tableaux numérotés et associatifs de personnes.

La suppression d'éléments

- *Tableau à une dimension*

Un tableau PHP est une collection d'informations, rangées les unes à côté des autres, accédées *via* un indice ou une étiquette. Chaque élément peut être supprimé individuellement par la fonction `unset()`. La syntaxe est :

```
unset($tab[1]);
```

où `$tab[1]` est l'indice de la case à supprimer dans le tableau `$tab`. Avec un tableau associatif, cela devient par exemple :

```
unset($tab['Nom']);
```

Remarque

La suppression d'un élément ne provoque aucun décalage des cases. Les éléments de `$tab` des cases 0, 2, 3... sont toujours présents, seule la case 1 et son contenu ont disparu.

Les fonctions `array_shift()` ou `array_pop()`suppriment respectivement le premier et le dernier élément. La fonction `array_shift()` décale les éléments vers la gauche. Ces fonctions retournent l'élément supprimé.

```
$premier_element = array_shift($tab_ent);//Décalage des cases
$dernier_element = array_pop($tab_noms) ;
```

- *Tableau à deux dimensions*

Numérotés

La suppression d'un élément dans un tableau à deux dimensions suit le même processus. La syntaxe suivante supprime l'élément situé à la ligne n° 1 et la colonne n° 0 du tableau `$tab2_ent`.

```
unset($tab2_ent[1][0]);
```

Associatifs

De la même manière, en indiquant les étiquettes des lignes/colonnes, on supprime un élément d'un tableau associatif. La syntaxe suivante supprime l'élément de la ligne `NICE` et de la colonne `CP` du tableau `$CP_DEPT`.

```
unset($CP_DEPT['NICE']['CP']);
```

Le parcours d'un tableau

Le parcours d'un tableau est le mécanisme utilisé pour traiter ses données ou effectuer un affichage contrôlé (à la place de `print_r`). Il utilise les boucles `for` pour les tableaux numérotés et `foreach` pour les tableaux associatifs.

- *Avec la boucle for*

Principe

Cette boucle a été présentée à la section 6.3 du chapitre 6. Elle est à privilégier pour les tableaux *numérotés* possédant des indices entiers.

Le parcours d'un tableau à une dimension utilise une boucle `for`, le parcours d'un tableau à deux dimensions utilise deux boucles `for` imbriquées, etc.

Avec une numérotation implicite (de 0 à *N*) ou explicite continue, par exemple de –10 à +10 avec tous les numéros, il est inutile de vérifier l'existence de la case. Avec une numérotation discontinue, comme avec les indices 1, 2, 5, 7, 10, il est indispensable de vérifier que la case existe avec la fonction `isset()`.

Tableau à une dimension

Le programme `tableau_parcours_for1_shell.php` affiche deux tableaux :

- `$tab_ent` est un tableau d'entiers numéroté en continu ;
- `$tab_noms` est un tableau de chaînes de caractères numéroté en discontinu.

La fonction `count()`retourne le nombre d'éléments du tableau, la fonction `reset()`positionne le pointeur interne au début du tableau, la fonction `end()`positionne le pointeur interne à la fin du tableau, la fonction `key()`retourne l'indice de la position courante. On récupère les numéros du premier et du dernier indice pour parcourir le tableau `$tab_noms` *via* la boucle `for`. Un test vérifie l'existence de la case `$i` avec `isset()`. Si c'est le cas, on affiche son contenu.

Listing 7.24 – Programme tableau_parcours_for1_shell.php

```
<?php
$tab_ent = array (18,22,-25,-3,-5,88); // tableau d'entiers
// Création d'un tableau de chaînes de caractères
$tab_noms = array (0=>'Dupont', 10=>'Martin', 20=>'Durand');
echo '--- Affichage de $tab_ent ---'.PHP_EOL;
$taille=count($tab_ent);
for ($i=0;$i<$taille;$i++)
{echo "[$i]:$tab_ent[$i]\t";}
echo PHP_EOL;
echo '--- Affichage de $tab_noms ---'.PHP_EOL;
reset($tab_noms); // Positionne le pointeur interne au début
$premier_indice=key($tab_noms); // récupère l'indice courant
end($tab_noms); // Positionne le pointeur interne à la fin
$dernier_indice=key($tab_noms); // récupère l'indice courant
reset($tab_noms); // Positionne le pointeur interne au début
for ($i=$premier_indice;$i<=$dernier_indice;$i++)
{if (isset($tab_noms[$i])) // Si l'élément existe
 {echo "[$i]:$tab_noms[$i]\t";}
}
echo PHP_EOL;
?>
```

L'exécution affiche l'indice entre `[]`, le caractère « : », puis la valeur de la case.

Listing 7.25 – Exécution de tableau_parcours_for1_shell.php

```
$ php tableau_parcours_for1_shell.php
--- Affichage de $tab_ent ---
[0]:18  [1]:22  [2]:-25  [3]:-3  [4]:-5  [5]:88
--- Affichage de $tab_noms ---
[0]:Dupont  [10]:Martin  [20]:Durand
```

Tableau à deux dimensions

Le programme `tableau_parcours_for2_shell.php` affiche deux tableaux à deux dimensions : `$tab2_ent` un tableau d'entiers ; `$tab2_noms` un tableau de chaînes de caractères. La particularité de `$tab2_ent` est que seule la dernière ligne possède trois colonnes. Deux boucles `for` affichent les lignes et les colonnes de ces tableaux. La fonction `isset()`vérifie l'existence de la case avant son affichage.

Listing 7.26 – Programme tableau_parcours_for2_shell.php

```
<?php
$tab2_ent=array (array(18,22),array(-25,-3),array(-5,88,120));
// Création d'un tableau de chaînes de caractères
$tab2_noms=array (array('Dupont', 'Martin'),
                  array('Durand','Mery'));
echo '--- Affichage de $tab2_ent ---'.PHP_EOL;
for ($i=0;$i<3;$i++)
{ for ($j=0;$j<3;$j++)
  {if (isset($tab2_ent[$i][$j]))
   {$valeur=$tab2_ent[$i][$j];
    echo "[$i][$j]:$valeur\t";
   }
  }
  echo PHP_EOL;
}
echo PHP_EOL;
echo '--- Affichage de $tab2_noms ---'.PHP_EOL;
for ($i=0;$i<2;$i++)
{ for ($j=0;$j<2;$j++)
  {if (isset($tab2_noms[$i][$j]))
   {$valeur=$tab2_noms[$i][$j];
    echo "[$i][$j]:$valeur\t";
   }
  }
echo PHP_EOL;
}
echo PHP_EOL;
?>
```

L'exécution affiche les indices entre `[]`, suivi de « : », puis la valeur de la case.

Listing 7.27 – Exécution de tableau_parcours_for2_shell.php

```
$ php tableau_parcours_for2_shell.php
--- Affichage de $tab2_ent ---
[0][0]:18   [0][1]:22
[1][0]:-25  [1][1]:-3
[2][0]:-5   [2][1]:88   [2][2]:120
--- Affichage de $tab2_noms ---
[0][0]:Dupont   [0][1]:Martin
[1][0]:Durand   [1][1]:Mery
```

- *Avec la boucle foreach*

Principe

La syntaxe générale de cette boucle a été présentée à la section 6.3 du chapitre 6. Elle est à privilégier pour les tableaux *associatifs*, donc avec des *étiquettes*, mais elle s'applique aussi aux tableaux numérotés.

Le parcours d'un tableau à une dimension utilise une boucle `foreach`, le parcours d'un tableau à deux dimensions utilise deux boucles `foreach` imbriquées, etc. Cette boucle récupère les valeurs et les étiquettes des cases.

Tableau à une dimension

Le programme `tableau_parcours_foreach1_shell.php` affiche deux tableaux à une seule dimension `$CP` et `$DEPT` présentés à la figure 7.8.

La première boucle `foreach` parcourt le tableau `$CP` et récupère le contenu de chaque case, dans la variable `$valeur` et l'affiche. La seconde boucle `foreach` parcourt le tableau `$DEPT` et récupère, pour chaque case, la valeur dans la variable `$valeur`, et l'étiquette dans la variable `$clef`, et affiche ces deux informations.

Listing 7.28 – Programme tableau_parcours_foreach1_shell.php

```
<?php
// Tableau des codes postaux
$CP = array ('CRETEIL' => 94000,'NICE'       => 6000,
             'LA BAULE'=> 44500,'STRASBOURG' => 67000);
// Tableau des départements d'appartenance
$DEPT=array('CRETEIL'    => 'Val-de-Marne',
            'NICE'       => 'Alpes-Maritimes',
            'LA BAULE'   => 'Loire-Atlantique',
            'STRASBOURG' => 'Bas-Rhin');
echo '--- Affichage de $CP ---'.PHP_EOL;
foreach($CP as $valeur)
{echo "$valeur\t";}
echo PHP_EOL;
echo '--- Affichage de $DEPT ---'.PHP_EOL;
foreach($DEPT as $clef => $valeur)
{echo "[$clef]:$valeur\t";}
echo PHP_EOL;
?>
```

Voici son exécution :

Listing 7.29 – Exécution de tableau_parcours_foreach1_shell.php

```
$ php tableau_parcours_foreach1_shell.php
--- Affichage de $CP ---
94000   6000   44500   67000
--- Affichage de $DEPT ---
[CRETEIL]:Val-de-Marne   [NICE]:Alpes-Maritimes
[LA BAULE]:Loire-Atlantique   [STRASBOURG]:Bas-Rhin
```

Tableau à deux dimensions

Les lignes de syntaxes suivantes affichent le tableau à deux dimensions `$CP_DEPT` qui a été présenté à la figure 7.9.

La première boucle `foreach` parcourt les lignes du tableau `$CP_DEPT`, récupère dans la variable `$codeville` l'étiquette de chaque ligne (`CRETEIL`, `NICE`…) et dans

la variable `$ligne_ville` le contenu de la ligne soit l'ensemble des colonnes de cette ligne (tableau à une dimension).

La seconde boucle `foreach` imbriquée parcourt les colonnes du tableau `$ligne_ville`, récupère dans la variable `$colonne` l'étiquette de chaque colonne (`CP`, `DEPT`), et dans la variable `$valeur` le contenu de la case traitée.

```
foreach($CP_DEPT as $codeville => $ligne_ville)
{ echo "$codeville: \t";
  foreach($ligne_ville as $colonne => $valeur)
  {echo "[$colonne]:$valeur\t";}
  echo PHP_EOL;
}
echo PHP_EOL;
```

- *La boucle while…current…next*

Principe

Cette section présente le parcours d'un tableau avec la boucle `while` et les fonctions `current()`et `next()`. Chaque itération progresse vers l'élément suivant *via* la fonction `next()`, tant que l'élément courant, retourné par `current()`existe (FALSE sinon). L'indice ou l'étiquette de la case est obtenu *via* la fonction `key()`.

Tableau à une dimension

Le programme `tableau_parcours_while_next1_shell.php` affiche deux tableaux à une seule dimension : `$tab_ent`, `$CP`.

- `$tab_ent` est un tableau numéroté continu d'entiers ;
- `$CP` est un tableau associatif de chaînes de caractères.

Listing 7.30 – Programme tableau_parcours_while_next1_shell.php

```
<?php
$tab_ent = array (18,22,-25,-3,-5,88); // Tableau d'entiers
// Tableau associatif des codes postaux
$CP = array ('CRETEIL'  => 94000,'NICE'       => 6000,
             'LA BAULE' => 44500,'STRASBOURG' => 67000);
echo '--- Affichage de $tab_ent ---'.PHP_EOL;
while ($valeur=current($tab_ent))
{$clef=key($tab_ent);
 echo "[$clef]:$valeur\t";
 next($tab_ent);
}
echo PHP_EOL;
echo '--- Affichage de $CP ---'.PHP_EOL;
while ($valeur=current($CP))
{$clef=key($CP);
 echo "[$clef]:$valeur\t";
 next($CP);
}
echo PHP_EOL;
?>
```

Voici son exécution :

Listing 7.31 – Exécution de tableau_parcours_while_next1_shell.php

```
--- Affichage de $tab_ent ---
[0]:18   [1]:22   [2]:-25   [3]:-3   [4]:-5   [5]:88
--- Affichage de $CP ---
[CRETEIL]:94000   [NICE]:6000   [LA BAULE]:44500   [STRASBOURG]:67000
```

Tableau à deux dimensions

Les lignes suivantes affichent le tableau associatif `$CP_DEPT` de la figure 7.9. Dans première boucle `while :` la fonction `current($CP_DEPT)` retourne le contenu de la première ligne dans la variable `$tabligne` ; la variable `$tabligne` est un tableau à une dimension contenant tous les éléments (colonnes) de la ligne. La fonction `key($CP_DEPT)` retourne le numéro de la ligne dans la variable `$numligne` ; la fonction `next($CP_DEPT)` fait progresser à la ligne suivante.

Dans la deuxième boucle `while` imbriquée : la fonction `current($tabligne)` retourne le contenu de la case de la colonne courante de cette ligne, dans la variable `$valeur` ; la fonction `key($tabligne)` retourne le numéro de la colonne dans la variable `$numcolonne` ; la fonction `next($tabligne)` fait progresser à la colonne suivante sur cette ligne.

```
while ($tabligne=current($CP_DEPT))
{$numligne=key($CP_DEPT);
 while ($valeur=current($tabligne))
 { $numcolonne=key($tabligne);
   echo "[$numligne][$numcolonne]:$valeur\t";
   next($tabligne);
 }
 echo PHP_EOL;
 next($CP_DEPT);
}
echo PHP_EOL;
```

La recherche dans un tableau

La recherche « classique » dans un tableau s'appuie sur une boucle de parcours incluant un test vérifiant si la valeur recherchée est celle de la case traitée, mais PHP propose également des fonctions spécifiques aux tableaux.

- *Boucle de recherche*

Cette section présente le mécanisme « classique » de recherche dans un tableau.

Tableau à une dimension

La recherche d'un élément ou d'une clef utilise les boucles vues précédemment avec un test qui vérifie si la donnée ou la clef est trouvée. Les syntaxes suivantes présentent la recherche d'une *valeur* dans un tableau associatif `$CP` et d'une *clef* dans un

tableau $DEPT, à partir de boucles foreach (tableaux de la figure 7.8). Une variable $trouve indique en sortie de boucle si l'élément a été trouvé.

```
echo "Entrez une donnée à chercher :";
fscanf(STDIN,"%d",$codepostal);
$trouve=false;
foreach($CP as $valeur) { // Recherche de la valeur
 if ($valeur == $codepostal) $trouve=true;
}
if ($trouve) echo "$codepostal trouvé".PHP_EOL;
else echo "$codepostal non trouvé".PHP_EOL;
echo "Entrez une clé à chercher :";
$clefrecherche=fgets(STDIN);
// Suppression des espaces (début,fin) et du saut de ligne
$clefrecherche=trim($clefrecherche);
$trouve=false;
foreach($DEPT as $clef => $valeur) { // Recherche de la clef
 if ($clef == $clefrecherche) $trouve=true;
}
if ($trouve) echo "$clefrecherche trouvé".PHP_EOL;
else echo "$clefrecherche non trouvé".PHP_EOL;
```

Remarque

La boucle foreach parcourt tout le tableau. Pour une recherche multi-occurrences, c'est la boucle à utiliser. Mais dans cet exemple, seule une occurrence est recherchée, une boucle while serait plus appropriée et permettrait d'arrêter le traitement dès l'élément trouvé.

Tableau à deux dimensions

Les syntaxes suivantes présentent la recherche d'une *valeur* et d'une *clef* dans un tableau $CP_DEPT à deux dimensions (figure 7.9) à partir de boucles foreach.

```
echo "Entrez une donnée à chercher :";
$donneerecherche = fgets(STDIN);
$donneerecherche = trim($donneerecherche);
echo "Entrez une clé à chercher :";
$clefrecherche = fgets(STDIN);
// Suppression des espaces (début,fin) et du saut de ligne
$clefrecherche = trim($clefrecherche);
$trouvedonnee  = false;
$trouveclef    = false;
// Boucle de recherche de la clef
// ... parcours des lignes
foreach($CP_DEPT as $codeville => $ligne_ville)
{ // Test si la clef recherchée est trouvée à cette ligne
  if ($codeville == $clefrecherche)
  { $trouveclef  = true   ;
    $coordonnees = "ligne";
  }
```

```
  // ... parcours des colonnes
  foreach($ligne_ville as $colonne => $valeur)
  { // Test si la clef est trouvée à cette colonne
    if ($colonne == $clefrecherche)
    {$trouveclef  = true     ;
     $coordonnees = "colonne";
    }
    // Test si la donnée est trouvée à cette ligne et colonne
    if ($valeur == $donneerecherche)
    {$trouvedonnee = true      ;
     $ligne        = $codeville;
    }
  }
}
if ($trouvedonnee)
  echo "La valeur $donneerecherche est trouvée à la ligne $ligne et la
  colonne $colonne".PHP_EOL;
else
  echo "La valeur $donneerecherche est non trouvée".PHP_EOL;
if ($trouveclef)
  echo "La clef $clefrecherche est trouvée en $coordonnees".PHP_EOL;
else echo "La clef $clefrecherche est non trouvée".PHP_EOL;
```

- *Recherche de la clef array_key_exists()*

Principe

Cette fonction recherche une *clef* dans un tableau à une dimension. Elle retourne la valeur TRUE si la clef est trouvée, et FALSE sinon. Sa syntaxe est :

```
$trouve = array_key_exists($clefrecherche,$TAB);
```

où `$trouve` est une variable booléenne, `$clefrecherche` la clef recherchée, et `$TAB` le tableau dans lequel la recherche est effectuée.

Parfois on vérifie l'existence d'une clef avec la fonction `isset()`. En réalité, on ne vérifie pas la clef, mais le contenu de la case. Dans le cas où la donnée de la case est la valeur null, `array_key_exists()` trouvera la clef, alors que `isset()` retournera faux, même si la clef existe car le contenu de la case est la valeur null.

Tableau à une dimension

Les syntaxes suivantes présentent la recherche d'une *clef* dans le tableau `$DEPT` (figure 7.8) *via* la fonction `array_key_exists()`.

```
echo "Entrez une clé à chercher : ";
$clefrecherche=fgets(STDIN);
$clefrecherche=trim($clefrecherche);
$trouve=false;
$trouve = array_key_exists($clefrecherche,$DEPT);
if ($trouve) echo "$clefrecherche trouvé".PHP_EOL;
else echo "$clefrecherche non trouvé".PHP_EOL;
```

Tableau à deux dimensions

Les lignes suivantes montrent l'utilisation de cette fonction sur le tableau à deux dimensions `$CP_DEPT` (figure 7.9). Cette fonction trouve les étiquettes des lignes mais pas des colonnes. Pour ces dernières, il faut utiliser une boucle `foreach` de parcours des lignes puis appliquer cette fonction à chaque tableau ligne extrait.

```
echo "Entrez une clé à chercher :";
$clefrecherche=fgets(STDIN);
$clefrecherche=trim($clefrecherche);
// Fonction array_key_exists() sur les étiquettes des lignes
$trouveclefligne=array_key_exists($clefrecherche,$CP_DEPT);
foreach($CP_DEPT as $codeville => $ligne_ville)
{//Fonction array_key_exists()sur les étiquettes des colonnes
 $trouveclefcol=array_key_exists($clefrecherche,$ligne_ville);
}
if ($trouveclefligne)
   echo "La clef $clefrecherche est trouvée dans les lignes".PHP_EOL;
if ($trouveclefcol)
   echo "La clef $clefrecherche est trouvée dans les colonnes".PHP_EOL;
if ((!$trouveclefligne) && (!$trouveclefcol))
   echo "La clef $clefrecherche est non trouvée".PHP_EOL;
```

- *Recherche des clefs array_keys()*

Principe

Cette fonction retourne toutes les clés ou un sous-ensemble de clés d'un tableau. Elle retourne un tableau de clefs. Ses différentes syntaxes sont :

```
$liste_clefs = array_keys($TAB);
$liste_clefs = array_keys($TAB,$valeur);
$liste_clefs = array_keys($TAB,$valeur,$strict);
```

où `$TAB` est le tableau contexte de la recherche, et `$valeur`, le contenu des cases pour lesquelles les clefs sont retournées. `$strict` est un booléen, si sa valeur est `true`, la donnée comparée avec `$valeur` doit être identique et du même type.

Tableau à une dimension

Les syntaxes suivantes présentent la recherche de toutes les clefs dans le tableau `$DEPT` (figure 7.8).

```
$liste_clefs = array_keys($DEPT);
print_r($liste_clefs);
$liste_clefs = array_keys($DEPT,'Val-de-Marne');
print_r($liste_clefs);
```

Le premier appel retourne toutes les clefs. Voici le résultat du `print_r()` :

```
Array
(
```

```
    [0] => CRETEIL
    [1] => NICE
    [2] => LA BAULE
    [3] => STRASBOURG
    [4] => SAINT MAUR
)
```

Le second appel retourne les clefs dont la donnée est `'Val-de-Marne'`, soit `'CRETEIL'` et `'SAINT MAUR'`. Voici le résultat du `print_r()` :

```
Array
(
    [0] => CRETEIL
    [1] => SAINT MAUR
)
```

Tableau à deux dimensions

Les lignes suivantes montrent l'utilisation de cette fonction sur le tableau à deux dimensions `$CP_DEPT` (figure 7.9). Cette fonction trouve les étiquettes des lignes mais pas des colonnes. Pour ces dernières, il faut utiliser une boucle `foreach` de parcours des lignes puis appliquer cette fonction à chaque tableau ligne extrait.

```
// Fonction array_keys() sur les étiquettes des lignes
$liste_clefs_lig = array_keys($CP_DEPT);
print_r($liste_clefs_lig);
foreach($CP_DEPT as $codeville => $ligne_ville)
{// Fonction array_keys() sur les étiquettes des colonnes
 $liste_clefs_col = array_keys($ligne_ville);
}
print_r($liste_clefs_col);
```

Le premier appel retourne les clefs des lignes. Le second appel retourne les clefs des colonnes.

- *Recherche d'une valeur in_array()*

Principe

Cette fonction retourne `true` si une donnée est dans le tableau. Ses syntaxes sont :

```
$trouve = in_array($valeur,$TAB);
$trouve = in_array($valeur,$TAB,$strict);
```

où `$TAB` est le tableau contexte de la recherche et `$valeur` la donnée recherchée. `$strict` est un booléen, si sa valeur est `true`, la donnée comparée avec `$valeur` doit être identique et du même type.

Tableau à une dimension

Les syntaxes suivantes présentent la recherche dans le tableau `$DEPT` (figure 7.8).

```
echo "Entrez une donnée à chercher :";
$donneerecherche=fgets(STDIN);
$donneerecherche=trim($donneerecherche);
$trouve=false;
$trouve = in_array($donneerecherche,$DEPT);
if ($trouve) echo "$donneerecherche trouvé".PHP_EOL;
else echo "$donneerecherche non trouvé".PHP_EOL;
```

Tableau à deux dimensions

Les lignes suivantes montrent l'utilisation de cette fonction sur le tableau à deux dimensions `$CP_DEPT` (figure 7.9).

```
echo "Entrez une donnée à chercher :";
$donneerecherche=fgets(STDIN);
$donneerecherche=trim($donneerecherche);
$trouve=false;
foreach($CP_DEPT as $codeville => $ligne_ville)
{ // Fonction in_array()
  $trouvedansligne = in_array($donneerecherche,$ligne_ville);
  if ($trouvedansligne) $trouve=true;
}
if ($trouve) echo "trouvé dans le tableau".PHP_EOL;
else echo "non trouvé dans le tableau".PHP_EOL;
```

- *Recherche des valeurs array_values()*

Principe

Cette fonction retourne toutes les données d'un tableau. Sa syntaxe est :

```
$liste_donnees = array_values($TAB);
```

où `$TAB` est le tableau contexte de la recherche.

Tableau à une dimension

Les syntaxes suivantes présentent la recherche de *toutes les données* dans le tableau `$DEPT` de la figure 7.8.

```
$liste_donnees=array_values($DEPT);
print_r($liste_donnees);
```

Voici le résultat obtenu pour le tableau `$DEPT`.

```
Array
(
    [0] => Val-de-Marne
    [1] => Alpes-Maritimes
    [2] => Loire-Atlantique
    [3] => Bas-Rhin
    [4] => Val-de-Marne
)
```

Tableau à deux dimensions

Les lignes suivantes montrent l'utilisation de cette fonction sur le tableau à deux dimensions `$CP_DEPT` de la figure 7.9. Une boucle `foreach` parcourt chaque ligne et applique cette fonction au tableau ligne extrait à chaque itération.

```
foreach($CP_DEPT as $codeville => $ligne_ville)
{ // Fonction array_values() sur chaque ligne
  $liste_donnees = array_values($ligne_ville);
  print_r($liste_donnees);
}
```

- *Recherche de la clef à partir de la valeur array_search()*

Principe

Cette fonction retourne la *clef associée à la donnée* recherchée dans le tableau. Ses syntaxes sont :

```
$clef = array_search($valeur,$TAB);
$clef = array_search($valeur,$TAB,$strict);
```

où `$TAB` est le tableau de recherche, `$valeur` la donnée recherchée. La variable `$strict` est un booléen. Si sa valeur est `true`, alors la comparaison avec `$valeur` doit être identique et du même type. La variable `$clef` contient l'étiquette ou le numéro de la case où est trouvée la première occurrence de `$valeur`, `false` sinon.

Tableau à une dimension

Les syntaxes suivantes présentent la recherche dans le tableau `$DEPT` (figure 7.8).

```
echo "Entrez une donnée à chercher :";
$donneerecherche=fgets(STDIN);
$donneerecherche=trim($donneerecherche);
$clef=array_search($donneerecherche,$DEPT);
if ($clef) echo "$donneerecherche trouvé en case $clef".PHP_EOL;
else echo "$donneerecherche  non trouvé".PHP_EOL;
```

Tableau à deux dimensions

Les lignes suivantes présentent cette fonction sur le tableau à deux dimensions `$CP_DEPT` de la figure 7.9.

```
echo "Entrez une donnée à chercher :";
$donneerecherche=fgets(STDIN);
$donneerecherche=trim($donneerecherche);
$colonne=false;
foreach($CP_DEPT as $codeville => $ligne_ville)
{ // Fonction array_search()
  $clefdansligne=array_search($donneerecherche,$ligne_ville);
  if ($clefdansligne) // On mémorise la ligne et la colonne
  {$ligne=$codeville;
   $colonne=$clefdansligne;
```

```
    }
  }
  if ($colonne)
    echo "$donneerecherche trouvé dans le tableau ligne=$ligne,
    colonne=$colonne".PHP_EOL;
  else
    echo "$donneerecherche non trouvé dans le tableau".PHP_EOL;
```

Gestion d'un tableau comme une pile ou file

Certains algorithmes utilisent des données « abstraites » appelées piles ou files.

Avec une *pile*, le dernier élément ajouté (empilé) est le premier retiré (dépilé). On parle de pile LIFO (*Last In*, *First Out*). Cela s'apparente à l'empilement de documents sur un bureau. C'est le document en haut de la pile (le dernier ajouté) qui est dépilé le premier.

Avec une *file*, le premier élément ajouté (enfilé) est le premier retiré (défilé), on parle de file FIFO (*First In*, *First Out*). Cela est similaire à une file d'attente au bureau de poste. La personne arrivée la première (la première de la file) passe la première (retirée de la file).

Les principales actions appliquées à une pile consistent à *empiler* une donnée (ajouter en fin de pile) ou *dépiler* une donnée (retirer de la fin de la pile). Les principales actions appliquées à une file consistent à *enfiler* une donnée (ajouter en fin de file) ou *défiler* une donnée (retirer du début de la file).

Lorsque ces données abstraites sont implémentées par des tableaux, le langage PHP propose les fonctions suivantes :

- `array_shift()` : retire un élément au début du tableau, c'est l'action défiler ;
- `array_unshift()` : ajoute un ou plusieurs éléments au début d'un tableau ;
- `array_push()` : ajoute un ou plusieurs éléments à la fin d'un tableau, c'est l'action empiler ou enfiler ;
- `array_pop()` : retire un élément de la fin d'un tableau, c'est l'action dépiler.

Des exemples d'utilisation des fonctions `array_shift()` et `array_pop()` ont été présentés à la section présentant la suppression d'éléments d'un tableau.

Les opérateurs sur les tableaux

Le tableau 7.4 récapitule les opérateurs sur les tableaux.

Tableau 7.4 – Opérateurs sur les tableaux

Opérateur	Signification	Exemple
+	Union	`$tab3 = $tab1 + $tab2 ;`
==	Égalité	`if ($tab1 == $tab2) ... ; /* vrai si tab1 et tab2 contiennent les mêmes paires clefs/valeurs */`

Opérateur	Signification	Exemple
===	Identique	`if ($tab1 === $tab2) ... ; /* vrai si tab1 et tab2 contiennent les mêmes paires clefs/valeurs dans le même ordre */`
!= ou <>	Inégalité	`if ($tab1 != $tab2) ... ; /* vrai si tab1 et tab2 ne sont pas égaux */`
!==	Non identique	`if ($tab1 !== $tab2) ... ; /* vrai si tab1 et tab2 ne sont pas identiques */`

Les fonctions sur les tableaux

Le tableau 7.5 présente quelques fonctions sur les tableaux. La liste complète est disponible en consultant l'URL : *https://secure.php.net/manual/fr/ref.array.php*.

Tableau 7.5 – Fonctions sur les tableaux

Fonction	Signification	Exemple
`array`	Crée un tableau	`$tab=array(18,22,-1);`
`array_column`	Retourne les valeurs d'une colonne d'un tableau	`$col=array_column($CP_DEPT,'DEPT');`
`array_count_values`	Compte le nombre de valeurs d'un tableau	`$tab=array_count_values($DEPT);`
`array_key_exists`	Vérifie si une clé existe dans un tableau	`$trouve=array_key_exists($clef,$tab);`
`array_keys`	Retourne toutes les clés ou un ensemble des clés	`$liste_clefs=array_keys($tab);`
`array_merge`	Fusionne plusieurs tableaux en un seul	`$tab=array_merge($T1,$T2);`
`array_pop`	Dépile un élément de la fin	`$der=array_pop($tab);`
`array_push`	Empile à la fin	`array_push($T,10,20);`
`array_reverse`	Inverse l'ordre	`$r=array_reverse($t);`
`array_search`	Recherche la clé associée à une valeur	`$clef=array_search($donnee,$tab);`
`array_shift`	Dépile un élément	`$pre=array_shift($t);`
`array_slice`	Extrait une portion de tableau	`$pt=array_slice($tab,2,3);`
`array_splice`	Efface et remplace une portion de tableau	`array_splice($t,2,4);`
`array_sum`	Calcule la somme des valeurs du tableau	`$s=array_sum($tab);`
`array_unshift`	Empile des éléments au début d'un tableau	`array_unshift($t,8,21);`
`array_values`	Retourne toutes les valeurs d'un tableau	`$liste_donnees=array_values($tab);`

Fonction	Signification	Exemple
compact	Crée un tableau à partir de variables et de leur valeur	$V="CRETEIL"; $CP=94000; $t=compact("V","CP");
count	Compte tous les éléments	$taille=count($tab);
current	Retourne l'élément courant	$val=current($tab) ;
end	Positionne à la fin	$val=end($tab);
explode	Coupe une chaîne en segments	$tab=explode(';',$chaine);
extract	Importe les variables dans la table des symboles	extract($DEPT); echo "$CRETEIL, $NICE,$LA_ BAULE\n";
implode	Assemble les éléments d'un tableau en une chaîne	$chaine_separateur = implode(";",$CP);
in_array	Indique si une valeur appartient à un tableau	$trouve=in_array($donnee,$tab);
is_array	Vrai si $tab est un tableau	$bl=is_array($tab);
key_exists	Alias de array_key_exists	$trouve=key_exists($clef,$tab);
key	Retourne une clé d'un tableau associatif	$indice=key($tab);
list	Assigne des variables à partir d'un tableau	list($v1,$v2)=$tab;
next	Avance le pointeur interne	$val=next($tab);
preg_split	Éclate une chaîne par expression rationnelle	$tab= preg_split("/[\s,]+/", "un, exemple");
prev	Recule le pointeur courant	$val=prev($tab);
print_r	Affiche un tableau	print_r($tab);
range	Crée un tableau contenant un intervalle d'éléments	$tab=range(0,12,2);
reset	Positionne au début	reset($tab);
rsort	Trie en ordre inverse	rsort($DEPT);
shuffle	Mélange les éléments	shuffle($tab);
sizeof	Alias de count	$taille=sizeof($tab);
sort	Trie un tableau	sort($DEPT);
split	Scinde une chaîne en un tableau, grâce à une expression rationnelle	$date="04/03/2015"; list($j,$m,$a)=split('[/.-]', $date);

7.4 Les fichiers

Définition

Un fichier est un espace de stockage sur disque. On y conserve les données qu'on désire retrouver lors d'un accès suivant. Le fichier peut également être une URL de la forme protocole://xxxxx.xxx, dans ce cas le langage PHP utilise le gestionnaire du protocole (ftp, http…) pour accéder aux données qui ne sont pas locales.

Les droits d'accès

L'accès à un fichier pose le problème du droit d'accès au fichier où au répertoire dans lequel il se trouve. Lors d'un accès web (*via* Apache), c'est l'utilisateur propriétaire du logiciel Apache sur le serveur qui accède au fichier. Cet utilisateur doit posséder les droits de lecture ou d'écriture selon le mode d'ouverture du fichier. Il faut alors préparer l'accès au fichier en affectant les « bons droits d'accès ».

Le type de fichier

Cette section présente les deux types fichiers, textes ou binaires.

Fichier texte

Le fichier est *lisible en dehors du programme PHP*. Il est constitué de lignes de texte terminées par un caractère fin de ligne, LF = `'\n'` sous Unix et CR = `'\r'` sous Windows. Les lectures et les écritures dans ce type de fichier sont généralement formatées, c'est-à-dire que le format des données lues ou écrites est précisé. Les instructions de lecture sont `fscanf()` ou `fread()`. Les instructions d'écriture sont `fprintf()` ou `fwrite()`.

Par défaut, sans précision particulière du mode d'ouverture *via* `fopen()`, le fichier sera de type texte. Il est possible de préciser explicitement qu'il s'agit d'un fichier texte en utilisant le mode « `t` » pour *texte* ou *traduction* de `fopen()`, accolé au mode « `r` » pour *lecture* ou « `w` » pour *écriture* : `'rt'`, ou `'wt'`.

Remarque

Pour éviter le problème de délimiteur de fin de ligne différent entre les fichiers textes Unix et Windows, il est possible d'utiliser uniquement le caractère `LF = '\n' (Unix)`, même sous Windows et d'activer le mode traduction `'t'` du fopen(). Dans ce cas, le caractère fin de ligne Unix sera traduit en caractère fin de ligne Windows.

Fichier binaire

Le fichier *n'est plus lisible en dehors du programme PHP*. Il est constitué d'une suite d'octets. Les lectures et les écritures gèrent alors des blocs d'octets. L'instruction de la lecture est `fread()`. L'instruction d'écriture est `fwrite()`.

L'ouverture d'un tel fichier par `fopen()` utilise le mode « `b` » pour *binary*, accolé au mode « `r` » pour *lecture* ou « `w` » pour *écriture* : `'rb'`, ou `'wb'`.

Les fichiers particuliers

Trois fichiers sont prédéfinis dans le langage PHP :

- `STDIN` qui correspond au clavier ;
- `STDOUT` qui correspond à l'écran ;
- `STDERR` qui correspond au canal des messages d'erreur, l'écran. Il peut ensuite être redirigé vers un autre périphérique que l'écran.

Ces fichiers sont ouverts en permanence. On les utilise en mettant leur nom (en majuscules) à la place de la variable fichier dans les instructions de lecture (`fscanf`) ou d'écriture (`fprintf`). Voici quelques exemples :

```
fprintf(STDOUT,"hello\n");
fprintf(STDERR,"message d'erreur\n");
echo "entrez un entier :";
fscanf(STDIN,"%d",$i);
```

Traitement des fichiers textes

L'ouverture fopen()

La liaison entre la variable `$fichier` et le nom réel du fichier, ou l'URL, se fait lors de l'ouverture du fichier *via* l'instruction `fopen()`. Sa syntaxe est de la forme :

```
$fichier = fopen(nom_du_fichier,'mode');
$fichier = fopen(url,'mode');
$fichier = fopen(nom_du_fichier,'mode',use_include_path);
```

où `$fichier` est la variable, `nom_du_fichier` est une chaîne de caractères indiquant le vrai nom du fichier, `'mode'` une chaîne de caractères précisant son mode d'ouverture, `use_include_path` est un paramètre optionnel booléen (à définir à `1` ou `true`) qui indique si le fichier doit être recherché dans le chemin de recherche (include_path) du système. Voici quelques exemples de syntaxe :

```
$fichier=fopen("/tmp/etud.txt",'r');     //Unix
$fichier=fopen("c:\\rep\\etud.txt","r"); //Windows
$fichier=fopen("/tmp/image.gif","wb");   //binaire
$fichier=fopen("http://www.exemple.fr","r"); //URL
```

Remarque

Pour Windows, on double le caractère « \ » ou bien on utilise le caractère « / » d'Unix.

Si le fichier existe, la variable `$fichier` pointe sur le fichier. Si le fichier est inaccessible (inexistant ou droits insuffisants), la variable `$fichier` contient la valeur

FALSE, et une alerte E_WARNING est générée. L'opérateur de contrôle d'erreur @ ajouté en préfixe ignore le message d'erreur. La syntaxe devient :

```
$fichier=@fopen("liste_eleves.txt",'r');
```

Le mode précise si le fichier doit être ouvert en lecture, écriture, ajout… Le tableau 7.6 présente les différents modes.

Tableau 7.6 – Modes d'ouverture d'un fichier

Mode	Signification	Exemple
`'r'`	Ouvre en lecture au début du fichier.	`$f=fopen("etud.txt","r");`
`'r+'`	Ouvre en lecture et écriture, et place le pointeur du fichier au début.	`$f=fopen("etud.txt","r+");`
`'w'`	Ouvre en écriture au début du fichier et réduit la taille à 0. Si le fichier n'existe pas, on tente de le créer.	`$f=fopen("etud.txt","w");`
`'w+'`	Ouvre en lecture et écriture, puis identique à « w ».	`$f=fopen("etud.txt","w+");`
`'a'`	Ouvre en écriture à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.	`$f=fopen("etud.txt","a");`
`'a+'`	Ouvre en lecture et écriture, puis identique à « a ».	`$f=fopen("etud.txt","a+");`
`'x'`	Crée et ouvre le fichier en écriture au début du fichier. Si le fichier existe déjà, **fopen()** échoue, et retourne `FALSE` en générant une erreur `E_WARNING`. Si le fichier n'existe pas, **fopen()** le crée.	`$f=fopen("etud.txt","x");`
`'x+'`	Crée et ouvre le fichier pour lecture et écriture, puis identique à « x ».	`$f=fopen("etud.txt","x+");`
`'c'`	Ouvre le fichier pour écriture. Si le fichier n'existe pas, il est créé, s'il existe, il n'est pas tronqué et l'appel à la fonction n'échoue pas. Le pointeur du fichier est au début. Ce mode est utile pour obtenir un verrou avant de tenter de modifier le fichier, car *'w'* tronque le fichier avant d'obtenir le verrou.	`$f=fopen("etud.txt","c");`
`'c+'`	Lecture et écriture, comportement identique au mode *'c'*.	`$f=fopen("etud.txt","c+");`

Remarques

L'ouverture d'un fichier binaire utilise le caractère 'b' ajouté à la fin des modes définis dans le tableau. Par exemple, l'ouverture d'un fichier binaire en lecture est 'rb', et l'ouverture d'un fichier binaire en écriture est 'wb'. Cela s'applique à tous les modes du tableau précédent.

Le mode d'ouverture d'un fichier texte avec traduction utilise le caractère 't' ajouté à la fin des modes définis dans le tableau. Par exemple, l'ouverture d'un fichier texte avec traduction en lecture est 'rt', et l'ouverture d'un fichier texte avec traduction en écriture est 'wt'. Cela s'applique à tous les modes du tableau précédent.

La fermeture fclose()

L'instruction `fclose()` ferme un fichier. Elle met à jour les informations sur le disque comme sa taille ou la liste des secteurs qu'il occupe. L'oubli de cette instruction implique que les données écrites dans le fichier, ne seront pas « référencées ». Ainsi la taille d'un tel fichier peut rester à 0. Sa syntaxe est de la forme :

```
fclose($fichier);
$retour=fclose($fichier);
```

où `$fichier` est la variable du fichier affectée par `fopen()`, et `$retour` est un booléen qui prend la valeur TRUE en cas de succès et FALSE sinon.

La lecture du fichier

Cette section présente les fonctions `fscanf()`, `fgetc()`, `fgets()` et `fread()`.

- *La fonction fscanf()*

L'instruction de lecture `fscanf()` effectue une *lecture formatée*. Le fichier doit être ouvert en mode lecture *via* `fopen()`. Sa syntaxe générale est :

```
fscanf($fichier,"%format",$var1,$var2,...);
$retour=fscanf($fichier,"%format",$var1,$var2,...);
$retour=fscanf($fichier,"%format"); //deux paramètres
```

où `$fichier` est la variable affectée par `fopen()`.`$retour` est un tableau si seulement deux paramètres sont passés à la fonction, le nombre de valeurs assignées dans les autres cas de lecture correcte ; FALSE en fin de fichier. `$var1,$var2,...` est la liste des variables recevant les données lues. `%format` est le format de lecture selon le type de la donnée (section 3.2 du chapitre 3). Voici des exemples de syntaxes :

```
$Tab_Info = fscanf($f1,"%s %s %s"); // retourne un tableau
$retour   = fscanf($f1,"%s %s",$nom,$prof);
```

Attention

Chaque `fscanf()` lit une ligne du fichier ! Les valeurs non lues de la ligne sont ignorées.

- *Les fonctions fgetc() et fgets()*

La fonction `fgetc()` lit un caractère et `fgets()` lit une ligne complète.

```
$caract = fgetc($f1) ; //Lit un caractère dans le fichier $f1
$nom    = fgets($f1) ; //Lit une chaîne dans le fichier $f1
```

- *La fonction fread()*

L'instruction `fread()` effectue une lecture « brute » non formatée du fichier. Elle lit un bloc d'octets. Le fichier doit être ouvert en lecture. Sa forme est :

```
$chaine = fread($fichier,$longueur);
```

`$fichier` est la variable affectée par `fopen()`, `$longueur` est le nombre d'octets à lire et `$chaine` est la chaîne lue (suite d'octets). Voici un exemple :

```
$contenu = fread($f1,100);
```

La lecture s'arrête, quand le nombre d'octets est lu, ou à la fin du fichier.

L'écriture du fichier

Cette section présente les fonctions `fprintf()`, `fputc()`, `fputs()` et `fwrite()`.

- *La fonction fprintf()*

L'instruction de lecture `fprintf()` effectue une *écriture formatée*. Le fichier doit être ouvert en mode écriture *via* `fopen()`. Sa syntaxe générale est :

```
fprintf($fichier,"%format",$var1,$var2,...);
$retour = fprintf($fichier,"%format",$var1,$var2,...);
```

où `$fichier` est la variable affectée par `fopen()`.`$retour` est la longueur de la chaîne écrite. `$var1,$var2,...` est la liste des variables à écrire. `%format` est le format d'écriture décrit dans le tableau 7.7. Voici deux exemples de syntaxes :

```
fprintf($f1,"%s %s %d\n",$nom,$prenom,$age);
$retour = fprintf($f1,"%s %s\n",$nom,$prenom);
```

Tableau 7.7 – Format d'écriture de fprintf()

Type de donnée	Exemple
entier	`fprintf($f1,"%d",$age);` `fprintf($f1,"%c",$num);// écrit un caractère`
caractère	`fprintf($f1,"%s",$initiale);`
réel	`fprintf($f1,"%f",$rayon);`
chaîne	`fprintf($f1,"%s",$nom);`

- *La fonction fputs()*

Le langage PHP propose une instruction spécifique à l'affichage d'une chaîne de caractères. Sa syntaxe est :

```
fputs($f1, $nom) ; // Ecrit une chaîne dans le fichier $f1
```

- *La fonction fwrite()*

L'instruction `fwrite()` effectue une écriture « brute » dans un fichier. Elle écrit un bloc d'octets. Le fichier doit être ouvert en écriture. Sa forme est :

```
$retour = fwrite($fichier, $chaine, $longueur);
```

où `$fichier` est la variable affectée par `fopen()`, `$longueur` est le nombre d'octets à écrire, `$chaine` est la chaîne contenant les données à écrire, `$retour` est un entier indiquant le nombre d'octets écrits ou FALSE si une erreur est arrivée. Voici deux exemples :

```
$nboctets = fwrite($f1,$contenu,100);
$nboctets = fwrite($f1,$contenu);
```

Exemples

- *Lecture dans un fichier*

Avec *fscanf()*

Le programme `fichier_lecture_utilisateurs_texte1_shell.php` lit une liste de personnes *via* `fscanf()` à partir du fichier `donnees_utilisateurs.txt`.

Listing 7.32 – Fichier donnees_utilisateurs.txt

```
Leonard   peintre       Italie  33
Kurosawa  cinéaste      Japon   44
Steve     informaticien USA     25
```

Ce programme utilise la syntaxe du `fscanf()` avec seulement deux arguments : les données sont retournées dans un tableau `$Tab_Info`.

Listing 7.33 – Programme fichier_lecture_utilisateurs_texte1_shell.php

```
<?php
$f1 = fopen("donnees_utilisateurs.txt", "r");
while ($Tab_Info=fscanf($f1,"%s\t%s\t%s\t%d"))
{list ($nom,$prof,$pays,$age) = $Tab_Info;
 echo "----------------------------".PHP_EOL;
 echo "Nom        : ".$nom.PHP_EOL;
 echo "Profession : ".$prof.PHP_EOL;
 echo "Pays       : ".$pays.PHP_EOL;
 echo "Age        : ".$age.PHP_EOL;
}
echo "----------------------------".PHP_EOL;
fclose($f1);
?>
```

Voici son exécution :

Listing 7.34 – Exécution de fichier_lecture_utilisateurs_texte1_shell.php

```
$ php fichier_lecture_utilisateurs_texte1_shell.php
----------------------------
Nom        : Leonard
Profession : peintre
Pays       : Italie
Age        : 33
```

```
----------------------------
Nom        : Kurosawa
Profession : cinéaste
Pays       : Japon
Age        : 44
----------------------------
Nom        : Steve
Profession : informaticien
Pays       : USA
Age        : 25
```

La boucle `while` précédente peut s'écrire avec un `scanf()` ayant plus de deux arguments. Les données lues sont affectées à `$nom`, `$prof`, `$pays`, `$age` :

```
while ($nbval=fscanf($f1,"%s %s %s %d",$nom,$prof,$pays,$age))
{ ... }
```

Avec *fread()*

Le programme `fichier_lecture_utilisateurs_texte_fread1_shell.php` lit une liste de personnes *via* `fread()` à partir du fichier `donnees_utilisateurs.txt` précédent. Les données « brutes » sont rangées dans `$contenu`. La fonction `filesize()`renvoie la taille du fichier en nombre d'octets.

Listing 7.35 – Programme fichier_lecture_utilisateurs_texte_fread1_shell.php

```
<?php
$NomF = "donnees_utilisateurs.txt";
$f1 = fopen($NomF, "r");
$contenu = fread($f1,filesize($NomF));
var_dump($contenu);
fclose($f1);
?>
```

Lors de son exécution, la fonction `var_dump()` montre que la chaîne fait 81 caractères, et que les caractères fin de ligne sont présents dans la chaîne `$contenu`.

Listing 7.36 – Exécution de fichier_lecture_utilisateurs_texte_fread1_shell.php

```
$ php fichier_lecture_utilisateurs_texte_fread1_shell.php
string(81) "Leonard   peintre   Italie   33
Kurosawa   cinéaste   Japon   44
Steve   informaticien   USA   25
"
```

- *Écriture dans un fichier*

Avec *fprintf()*

Le programme `fichier_ecriture_utilisateurs_texte1_shell.php` saisit une liste de personnes et la sauvegarde dans `donnees_utilisateurs2.txt`.

Listing 7.37 – Programme fichier_ecriture_utilisateurs_texte1_shell.php

```
<?php
$f1 = fopen("donnees_utilisateurs2.txt", "w");
// --- Saisie d'une liste de personnes ---
$saisie="saisie non vide";
$nb_personnes=0;
while (!empty($saisie))
{echo "Entrez un nom, une profession, un pays et un âge (ex : Dupont
Etudiant France 28) : ";
 $saisie=fgets(STDIN);
 // --- Traitement de la chaîne lue, ---
 // Suppression des espaces (début,fin) et du saut de ligne
 $saisie=trim($saisie);
 // --- Remplace les espaces multiples par un seul espace ---
 $saisie= preg_replace('/\s{2,}/',' ',$saisie);
 // --- On vérifie que la saisie n'est pas vide ---
 if (!empty($saisie))
 {// Rangement dans les variables
  list($nom,$prof,$pays,$age)=explode(' ',$saisie);
  // --- Écriture dans le fichier ---
  fprintf($f1,"%s\t%s\t%s\t%d\n",$nom,$prof,$pays,$age);
  $nb_personnes++;
 }
}
echo "--- Sauvegarde dans donnees_utilisateurs2.txt : $nb_personnes
personnes---".PHP_EOL;
fclose($f1);
?>
```

Voici un exemple d'exécution :

Listing 7.38 – Exécution de fichier_ecriture_utilisateurs_texte1_shell.php

```
$ php fichier_ecriture_utilisateurs_texte1_shell.php
Entrez un nom, une profession, un pays et un âge (ex : Dupont Etudiant
France 28) : Dupont Etudiant France 28
Entrez un nom, une profession, un pays et un âge (ex : Dupont Etudiant
France 28) : Martin Ingenieur Allemagne 60
Entrez un nom, une profession, un pays et un âge (ex : Dupont Etudiant
France 28) :
--- Sauvegarde dans donnees_utilisateurs2.txt : 2 personnes---
```

Voici le fichier généré donnees_utilisateurs2.txt :

Listing 7.39 – Fichier donnees_utilisateurs2.txt

```
$ cat donnees_utilisateurs2.txt
Dupont  Etudiant   France     28
Martin  Ingenieur  Allemagne  60
```

Avec *fwrite()*

Le programme suivant saisit une liste de personnes et la sauvegarde dans le fichier `donnees_utilisateurs3.txt`. La sauvegarde écrit des octets. Les syntaxes identiques au programme précédent sont remplacées par des « ... ».

Listing 7.40 – Programme fichier_ecriture_utilisateurs_texte_fwrite1_shell.php

```
<?php
$f1 = fopen("donnees_utilisateurs3.txt", "w");
// --- Saisie d'une liste de personnes ---
$saisie="saisie non vide";
$totnboctets=0;
while (!empty($saisie))
{...
 if (!empty($saisie))
 {...
  // Écriture dans le fichier
  $contenu=$nom."\t".$prof."\t".$pays."\t".$age.PHP_EOL;
  $nboctets=fwrite($f1,$contenu,strlen($contenu));
  $totnboctets+=$nboctets;
 }
}
echo "--- Sauvegarde dans donnees_utilisateurs3.txt : $totnboctets
octets---".PHP_EOL;
fclose($f1);
?>
```

Traitement des fichiers binaires

Les fichiers binaires ne contiennent qu'une suite d'octets, sans aucun formatage. Ces fichiers sont illisibles en dehors d'un programme.

L'ouverture et la fermeture *fopen()* et *fclose()*

La syntaxe du `fopen()` pour les fichiers binaires est identique à celle des fichiers textes. Seul le mode d'ouverture change, avec l'ajout du `'b'`.

```
$fichier=fopen("/tmp/etud.data","rb") ; // Lecture binaire
$fichier=fopen("/tmp/image.gif","wb") ; // Écriture binaire
```

La syntaxe du `fclose()`est identique à celle des fichiers textes.

La lecture et l'écriture du fichier *fread()* et *fwrite()*

Les instructions `fread()` et `fwrite()` ont été présentées avec les fichiers textes. Leur syntaxe est identique, seul le contenu de la chaîne de caractères représentant le bloc d'octets change ! On compacte des données dans la chaîne « binaire » avec la fonction `pack()`. La fonction `unpack()` effectue le traitement inverse.

Compactage dans une chaîne binaire *pack()*

L'instruction `pack()` compacte les arguments dans une chaîne de caractères, selon le format indiqué au tableau 7.8. Sa syntaxe générale est :

```
$chaine=pack($format,$var1,$var2,...);
```

où `$chaine` est la chaîne de caractère recevant la suite d'octet. `$var1`, `$var2`… les variables contenant les données à compacter. Voici trois exemples de syntaxes :

```
$chaine   = pack("ifaa*",$i,$x,$lettre,$texte);
$chaine   = pack("C",$donnee);
$BOM_UTF8 = pack("C3",0xef,0xbb,0xbf);
```

Le *format* est une chaîne de caractères constituée de codes (*cf.* tableau 7.8) suivis par un *caractère répéteur* optionnel qui peut être :

- un entier (1, 2…) pour préciser le nombre d'arguments qui utilisent ce format ;
- * pour une répétition jusqu'à la fin des arguments ;
- pour les formats a, A, h, H, le répéteur indique combien de caractères de la donnée sont pris ;
- pour le format @ la valeur du répéteur indique la position absolue où l'on insère les prochaines données.

Tableau 7.8 – Format de pack()

Code	Description
a	Une chaîne terminée par NULL
A	Une chaîne terminée par un espace
h	Une chaîne en hexadécimal avec le bit de poids faible en premier
H	Une chaîne en hexadécimal avec le bit de poids fort en premier
c	Caractère signé
C	Caractère non signé
s	Entier court signé (16 bits, ordre des bits selon la machine)
S	Entier court non signé (16 bits, ordre des bits selon la machine)
i	Entier signé (taille et ordre des bits dépendants de la machine)
I	Entier non signé (taille et ordre des bits dépendants de la machine)
l	Entier long signé (32 bits, ordre des bits selon la machine)
L	Entier long non signé (32 bits, ordre des bits selon la machine)
N	Entier long non signé (toujours 32 bits, ordre des bits big endian)
V	Entier long non signé (toujours 32 bits, ordre des bits little endian)
q	Entier doublement long signé (toujours 64 bits, ordre des bits selon la machine)
Q	Entier doublement long non signé (toujours 64 bits, ordre des bits selon la machine)

Code	Description
J	Entier doublement long non signé (toujours 64 bits, ordre des bits big endian)
P	Entier doublement long non signé (toujours 64 bits, ordre des bits little endian)
f	Nombre à virgule flottante (taille et représentation selon la machine)
d	Nombre à virgule flottante double (taille et représentation dépendantes de la machine)
x	Caractère NUL
X	Recule d'un caractère
Z	Chaîne complétée par la valeur NULL (nouveau en PHP 5.5)
@	Remplit avec des NUL jusqu'à la position absolue

Décompactage à partir d'une chaîne binaire *unpack()*

L'instruction `unpack()` décompacte les octets d'une chaîne binaire, selon le format indiqué. Sa syntaxe générale est :

```
$tab=unpack($format,$chaine);
```

où `$chaine` est la chaîne binaire contenant les octets et `$tab` est un tableau de résultat. Voici quatre exemples de syntaxes :

```
$tab = unpack("ient/freel/acar/a*ch",$ChOctets);
$tab = unpack("C*entier",$ChOctets)            ;
$tab = unpack("C*",$ChOctets)                  ;
$tab = unpack("a7nom",$ChOctets)               ;
```

Le *format* est une chaîne de caractères identique à celle de `pack()`, suivi du *nom de la case* du tableau dans laquelle est rangée la donnée. Voici le contenu du tableau résultat selon les formats précédents :

- `"ient/freel/acar/a*ch"` : le tableau `$tab` possède quatre cases, nommées `ent`, `reel`, `car` et `ch`, respectivement typées en integer, float, string(1), string ;
- `"C*entier"` : le tableau `$tab` possède N cases entières, nommées `entierx`, où `x` est une valeur 1, 2… ;
- `"C*"` : `$tab` possède N cases entières, numérotées 1, 2… ;
- `"a7nom"` : `$tab` possède une case de type string(7), nommée `nom`.

Exemples

- *Écriture avec pack() et fwrite()*

Écriture d'une suite d'entiers

Le programme suivant écrit une suite d'entiers, dont la valeur est comprise entre 0 et 255, dans le fichier binaire `donnees_binaires.data`. Les valeurs sont : 0x10, 0x59, 0x42, 18, 34, 255, 200 soit 16, 89, 66, 18, 34, 255, 200.

Listing 7.41 – Programme fichier_ecriture_entiers_bin_pack_fwrite_shell.php.

```
<?php
$NomF="donnees_binaires.data";
// Tableau contenant les entiers compris entre 0 et 255
// Chaque entier peut être codé sur 1 octet
$tab_donnees_binaires=array(0x10,0x59,0x42,18,34,255,200);
foreach ($tab_donnees_binaires as $donnee)
{// Compactage des octets dans une chaîne d'octets
 // Ajout de l'entier au format C=caractère non signé=1 octet
 $ChOctets .= pack("C", $donnee);
}
// Écriture dans le fichier binaire
$fichier_binaire = fopen($NomF,'wb');
fwrite($fichier_binaire,$ChOctets);
fclose($fichier_binaire);
$nboctets=strlen($ChOctets);
echo "écriture dans $NomF de $nboctets octets\n";
?>
```

Voici son exécution :

Listing 7.42 – Exécution de fichier_ecriture_entiers_bin_pack_fwrite_shell.php.

```
$ php fichier_ecriture_entiers_bin_pack_fwrite_shell.php
écriture dans donnees_binaires.data de 7 octets
```

La commande Unix « `ls -l` » montre que la taille de `donnees_binaires.data` est bien de 7 octets :

```
$ ls -l donnees_binaires.data
-rw-r--r--  1 lery  501  7 19 oct 13:58 donnees_binaires.data
```

Écriture de données de différents types

Le programme suivant écrit : un entier ayant la valeur –1 024 ; un réel dont la valeur est –38.74 ; une chaîne d'un caractère `"A"` ; une chaîne de plusieurs caractères dont la valeur est `"Bonjour"`, dans le fichier binaire `donnees_bin_entier_reel_caractere_chaine.data`.

Listing 7.43 – Programme fichier_ecriture_donnees_bin_pack_fwrite_shell.php.

```
<?php
 $NomF="donnees_bin_entier_reel_caractere_chaine.data";
 // Variables contenant les données
 $i      = -1024    ;
 $x      = -38.74   ;
 $lettre = "A"      ;
 $texte  = "Bonjour";
 // Compactage des octets dans une chaîne d'octets
 $ChOctets = pack("ifaa*",$i,$x,$lettre,$texte);
 // Écriture dans le fichier binaire
```

```
 $fichier_binaire = fopen($NomF, 'wb');
 fwrite($fichier_binaire, $ChOctets);
 fclose($fichier_binaire);
 $nboctets=strlen($ChOctets);
 echo "écriture dans $NomF de $nboctets octets\n";
?>
```

Écriture d'une liste de personnes

Le programme suivant reprend le programme `fichier_ecriture_utilisateurs_texte1_shell.php` qui saisit une liste de personnes, et la sauvegarde dans le fichier binaire `donnees_utilisateurs.data`. Les lignes identiques au programme précédent sont remplacées par des « … ». Chaque personne est caractérisée par : son nom (chaîne) ; sa profession (chaîne) ; son pays (chaîne) ; son âge (entier). Afin de relire les chaînes de caractères, leur taille est stockée sous la forme d'un octet (taille limitée à 255 caractères) avant leur écriture. L'âge est stocké sur un octet.

Listing 7.44 – Programme fichier_ecriture_utilisateurs_bin_pack_fwrite_shell.php.

```
<?php
$NomF="donnees_utilisateurs.data";
$fichier_binaire = fopen($NomF, "wb");
// Saisie d'une liste de personnes
$saisie="saisie non vide";
$nb_personnes=0;
$ChOctets="";
while (!empty($saisie))
{// Saisie des noms, professions, pays et âges des personnes
 ...
 if (!empty($saisie))
 {// Rangement dans les variables
  list($nom,$prof,$pays,$age)=explode(' ',$saisie);
  // Compactage des octets dans une chaîne d'octets
  $taille=strlen($nom)+1;
  $ChOctets .= pack("c",$taille);
  $ChOctets .= pack("a{$taille}",$nom);
  $taille=strlen($prof)+1;
  $ChOctets .= pack("c",$taille);
  $ChOctets .= pack("a{$taille}",$prof);
  $taille=strlen($pays)+1;
  $ChOctets .= pack("c",$taille);
  $ChOctets .= pack("a{$taille}",$pays);
  $ChOctets .= pack("c",$age);
  $nb_personnes++;
 }
}
echo "--- Sauvegarde dans $NomF : $nb_personnes personnes ---".PHP_EOL;
// -- écriture --
fwrite($fichier_binaire, $ChOctets);
fclose($fichier_binaire);
?>
```

- *Lecture avec fread() et unpack()*

Lecture d'une suite d'entiers

Le programme suivant lit une suite d'entiers, dont la valeur est comprise entre 0 et 255, à partir du fichier binaire `donnees_binaires.data` produit par le programme `fichier_ecriture_entiers_bin_pack_fwrite_shell.php`. La fonction `unpack()` avec le format « `C*` » décompacte les octets de la chaîne `$ChOctets` et retourne le résultat dans le tableau `$tab_donnees_binaires`.

Listing 7.45 – Programme fichier_lecture_entiers_bin_unpack_fread_shell.php.

```
<?php
 $NomF="donnees_binaires.data";
 $f1 = fopen($NomF, "rb");
 // Lecture à partir du fichier binaire
 $ChOctets = fread($f1,filesize($NomF));
 // Décompacte la chaîne d'octets lue
 // au format C = caractère non signé = 1 octet
 // * indique que ce format s'applique à tous les octets
 $tab_donnees_binaires = unpack("C*",$ChOctets);
 // Boucle d'extraction des données du tableau
 foreach ($tab_donnees_binaires as $indice => $donnee)
 {echo "$indice : $donnee\n";}
 fclose($f1);
?>
```

Voici son exécution :

Listing 7.46 – Exécution de fichier_lecture_entiers_bin_unpack_fread_shell.php.

```
$ php fichier_lecture_entiers_bin_unpack_fread_shell.php
1 : 16
2 : 89
3 : 66
4 : 18
5 : 34
6 : 255
7 : 200
```

Lecture de données de différents types

Le programme suivant lit les données binaires produites par le programme `fichier_ecriture_donnees_bin_pack_fwrite_shell.php` soit : un entier (-1024) ; un réel (-38.74) ; une chaîne d'un caractère («A») ; une chaîne de plusieurs caractères («Bonjour»).

Le format `"ientier/freel/acaractere/a*chaine"` de la fonction `unpack()` décompacte les octets de `$ChOctets` et retourne le résultat dans le tableau associatif `$tab_donnees_binaires` dont les étiquettes de cases sont : « `entier ` », « `reel` », « `caractere` », « `chaine` ».

Listing 7.47 – Programme fichier_lecture_donnees_bin_unpack_fread_shell.php.

```
<?php
 // Lit un fichier, et le place dans une chaîne
 $NomF="donnees_bin_entier_reel_caractere_chaine.data";
 $f1 = fopen($NomF, "rb");
 // Lecture à partir du fichier binaire
 $ChOctets = fread($f1,filesize($NomF));
 // Décompactage des octets à partir de la  chaîne lue
 $tab_donnees_binaires = unpack("ientier/freel/acaractere/
 a*chaine",$ChOctets);
 // Boucle d'extraction des données du tableau
 foreach ($tab_donnees_binaires as $indice => $donnee)
 {echo "$indice : $donnee\n";}
 // Fermeture
 fclose($f1);
?>
```

Voici son exécution :

Listing 7.48 – Exécution de fichier_lecture_donnees_bin_unpack_fread_shell.php.

```
$ php fichier_lecture_donnees_bin_unpack_fread_shell.php
entier : -1024
reel : -38.740001678467
caractere : A
chaine : Bonjour
```

Lecture d'une liste de personnes

Le programme suivant lit les données binaires produites par le programme `fichier_ecriture_utilisateurs_bin_pack_fwrite_shell.php`. Le fichier binaire `donnees_utilisateurs.data` à lire est constitué d'une liste de personnes. Pour chaque personne, il faut lire les données suivantes :

- un octet : taille en entier de la chaîne nom ;
- un nom : chaîne de caractères (taille donnée par l'octet précédent) ;
- un octet : taille en entier de la chaîne profession ;
- une profession : chaîne de caractères (taille donnée par l'octet précédent) ;
- un octet : taille en entier de la chaîne pays ;
- un pays : chaîne de caractères (taille donnée par l'octet précédent) ;
- un octet : âge de la personne ;

Listing 7.49 – Programme fichier_lecture_utilisateurs_bin_unpack_fread_shell.php.

```
<?php
 $NomF="donnees_utilisateurs.data";
 $fichier_binaire = fopen($NomF, "rb");
 // Lecture à partir du fichier binaire
 $ChOctets = fread($fichier_binaire, filesize($NomF));
```

```
// Traitement de la chaine binaire lue
while (strlen($ChOctets) >0)
{// Décompactage des octets à partir de la  chaîne lue
 // -- Traitement du nom ---
 // On récupère un octet (entier)=nombre de caractères à lire
 $tab_donnees_binaires = unpack("ctaille",$ChOctets);
 $taille=$tab_donnees_binaires['taille'];
 // On prépare le format de lecture de la chaîne
 $format="a".$taille."nom";
 // On supprime l'octet (nb caractères à lire) de la chaîne
 $ChOctets=substr($ChOctets,1,strlen($ChOctets)-1);
 // On extrait la chaîne de caractères
 $tab_donnees_binaires = unpack($format,$ChOctets);
 $nom=$tab_donnees_binaires['nom'];
 // On supprime la donnée lue de la chaîne binaire
 $ChOctets=substr($ChOctets,$taille,strlen($ChOctets)-$taille);
 // -- Traitement de la profession ---
 // On récupère un octet (entier)=nombre de caractères à lire
 $tab_donnees_binaires = unpack("ctaille",$ChOctets);
 $taille=$tab_donnees_binaires['taille'];
 // On prépare le format de lecture de la chaîne
 $format="a".$taille."profession";
 // On supprime l'octet (nb caractères à lire) de la chaîne
 $ChOctets=substr($ChOctets,1,strlen($ChOctets)-1);
 // On extrait la chaîne de caractères
 $tab_donnees_binaires = unpack($format,$ChOctets);
 $prof=$tab_donnees_binaires['profession'];
 // On supprime la donnée lue de la chaîne binaire
 $ChOctets=substr($ChOctets,$taille,strlen($ChOctets)-$taille);
 // -- Traitement du pays ---
 // On récupère un octet (entier)=nombre de caractères à lire
 $tab_donnees_binaires = unpack("ctaille",$ChOctets);
 $taille=$tab_donnees_binaires['taille'];
 // On prépare le format de lecture de la chaîne
 $format="a".$taille."pays";
 // On supprime l'octet (nb caractères à lire) de la chaîne
 $ChOctets=substr($ChOctets,1,strlen($ChOctets)-1);
 // On extrait la chaîne de caractères
 $tab_donnees_binaires = unpack($format,$ChOctets);
 $pays=$tab_donnees_binaires['pays'];
 // On supprime la donnée lue de la chaîne binaire
 $ChOctets=substr($ChOctets,$taille,strlen($ChOctets)-$taille);
 // -- Traitement de l'âge ---
 // On récupère un octet (entier) = âge
 $tab_donnees_binaires = unpack("cage",$ChOctets);
 $age=$tab_donnees_binaires['age'];
 // On supprime la donnée lue de la chaîne binaire
 $ChOctets=substr($ChOctets,1,strlen($ChOctets)-1);
 // -- Affichage --
 echo "----------------------------".PHP_EOL;
 echo "Nom        : ".$nom.PHP_EOL;
 echo "Profession : ".$prof.PHP_EOL;
 echo "Pays       : ".$pays.PHP_EOL;
```

```
  echo "Age         : ".$age.PHP_EOL;
 }
 fclose($fichier_binaire);
?>
```

Voici son exécution :

Listing 7.50 – Exécution de fichier_lecture_utilisateurs_bin_unpack_fread_shell.php.

```
$ php fichier_lecture_utilisateurs_bin_unpack_fread_shell.php
----------------------------
Nom         : Dupont
Profession  : Etudiant
Pays        : France
Age         : 28
----------------------------
Nom         : Martin
Profession  : Ingenieur
Pays        : Allemagne
Age         : 60
----------------------------
Nom         : Durand
Profession  : Cadre
Pays        : Italie
Age         : 44
```

Fonctions sur les fichiers

Le tableau 7.9 présente quelques fonctions sur les fichiers. La liste complète est accessible à l'URL : *http://php.net/manual/fr/ref.filesystem.php*.

Tableau 7.9 – Fonctions sur les fichiers

Fonction	Signification	Exemple
basename	Retourne le nom du fichier dans un chemin	`$nom=basename($NomF,".data");`
dirname	Renvoie le chemin du dossier parent	`$rep=dirname($chemin);`
fclose	Ferme un fichier	`fclose($f);`
feof	Teste la fin du fichier	`while (!feof($f)) ...`
fflush	Envoie tout le contenu généré dans un fichier	`fflush($file);`
fgetc	Lit un caractère	`$car = fgetc($f)`
fgets	Récupère la ligne courante	`$ch=fgets($f,200) ;`
file_exists	Vérifie si un fichier ou un dossier existe	`$res=file_exists($NomF);`
file_get_contents	Lit tout un fichier dans une chaîne	`$ch = file_get_contents($F);`
file_put_contents	Écrit un contenu dans un fichier	`file_put_contents($NomF,$tab);`

Fonction	Signification	Exemple
file	Lit le fichier et renvoie le résultat dans un tableau	`$tab = file($NomF);`
filesize	Lit la taille d'un fichier	`$nb=filesize($NomF) ;`
filetype	Renvoie le type du fichier	`$type=filetype($NomF);`
fopen	Ouvre un fichier	`$f=fopen($NomF,"r");`
fputcsv	Formate une ligne en CSV et l'écrit	`foreach($tab as $champ) {fputcsv($f, $champ);}`
fread	Lecture en mode binaire	`$contenu=fread($f,100);`
fscanf	Lecture du fichier en mode texte	`fscanf($f,"%s",$nom) ;`
fseek	Modifie la position du pointeur de fichier	`fseek($f,0);`
ftell	Renvoie la position courant du pointeur de fichier	`$pos=ftell($f);`
ftruncate	Tronque un fichier	`ftruncate($f,$taille) ;`
fwrite	Écrit un fichier en mode binaire	`$nb=fwrite($f,$ch,strlen($ch));`
is_dir	Test si c'est un dossier	`$res=is_dir("./Rep");`
is_executable	Indique si le fichier est exécutable	`$res=is_executable($NomF);`
is_file	Indique si le fichier est un véritable fichier	`$res=is_file($NomF);`
is_link	Indique si le fichier est un lien symbolique	`$res=is_link($NomF);`
is_readable	Vrai si le fichier existe et est accessible en lecture	`$res=is_readable($NomF);`
is_writable	Vrai si le fichier est accessible en écriture	`$res=is_writable($NomF);`
parse_url	Analyse une URL et retourne ses composants	`$URL = 'http://lery:pdp@ser.dom.fr/chemin?argument=valeur';` `print_r(parse_url($URL));`
pathinfo	Retourne des informations sur un chemin système	`$elem_chemin=pathinfo("./donnees_utilisateurs.data");` `print_r($elem_chemin);`
readfile	Affiche un fichier	`readfile($file);`
rewind	Replace le pointeur de fichier au début	`rewind($file);`
stat	Renvoie les informations à propos d'un fichier	`$tabstat=stat($NomF);`
tempnam	Crée un fichier avec un nom unique	`$NomF=tempnam(".","temp");`
tmpfile	Crée un fichier temporaire	`$temp=tmpfile();`
unlink	Efface un fichier	`$res=unlink($Nouveau);`

Exercices

7.1 Faire un programme qui saisit puis qui affiche une liste de personnes dans un environnement web. Les personnes sont saisies *via* un formulaire (figure 7.13).

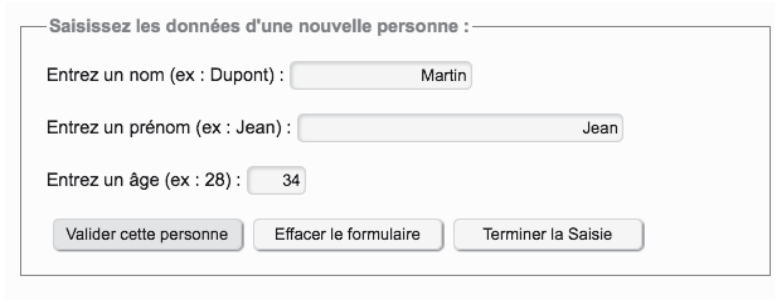


Figure 7.13 – **Gestion d'une liste de personnes : formulaire de saisie.**

À chaque validation, le formulaire de saisie est à nouveau affiché, suivi d'un récapitulatif de la personne précédemment validée (figure 7.14 gauche) ou des éventuels messages d'erreurs liés aux champs vides (figure 7.14 droite).

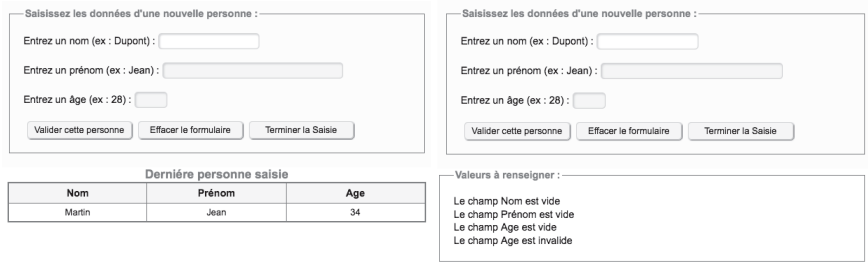


Figure 7.14 – **Gestion d'une liste de personnes : nouvelle saisie.**

Le bouton « Terminer la saisie » clôture la saisie et affiche la liste des personnes (figure 7.15).

Tableau des personnes

ID	Nom	Prénom	Age
0	Martin	Jean	34
1	De-La-Rue	Jean-Christophe	54
2	Dupont	Pierre	45

Figure 7.15 – **Gestion d'une liste de personnes : synthèse des personnes.**

Les espaces en début et en fin de saisie sont supprimés. Les espaces multiples sont remplacés par un seul espace. Les noms et prénoms, y compris composés, sont

normalisés en minuscules avec la première lettre majuscule et un « - » de liaison remplace l'espace de séparation restant pour les noms et prénoms composés.

7.2 Faire un programme qui reprend l'exercice précédent et ajoute la sauvegarde, dans un fichier texte, des personnes saisies. Les personnes sont saisies *via* un formulaire (figures 7.13 et 7.14). Le bouton « Terminer la saisie » clôture la saisie, affiche la liste des personnes et saisit le nom du fichier (figure 7.16).

Tableau des personnes

ID	Nom	Prénom	Age
0	Martin	Jean	34
1	De-La-Rue	Jean-Christophe	54
2	Dupont	Pierre	45

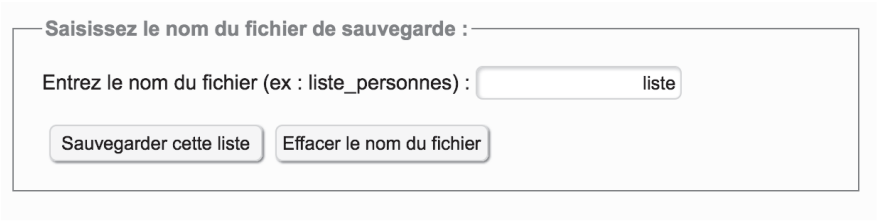


Figure 7.16 – Sauvegarde d'une liste de personnes : saisie du nom du fichier.

Le bouton « Sauvegarder cette liste » affiche un écran de confirmation avec le nom du fichier créé dans le sous-répertoire « Sauvegardes ». L'extension « .txt » est ajoutée, ou remplace l'extension saisie. Un tableau récapitulatif des personnes sauvegardées est affiché avec leur nombre (figure 7.17).

Message :

Sauvegarde dans le fichier **Sauvegardes/liste.txt** :

Tableau des personnes sauvegardées

ID	Nom	Prénom	Age
0	Martin	Jean	34
1	De-La-Rue	Jean-Christophe	54
2	Dupont	Pierre	45

Nombre de personnes sauvegardées : 3

Figure 7.17 – Sauvegarde d'une liste de personnes : confirmation.

En cas d'erreur de sauvegarde, l'écran de la figure 7.18 apparaît et présente : un message avec le nom du fichier ; l'interprétation des erreurs émises par le système ; le nom du programme PHP et la ligne ayant rencontré l'erreur.

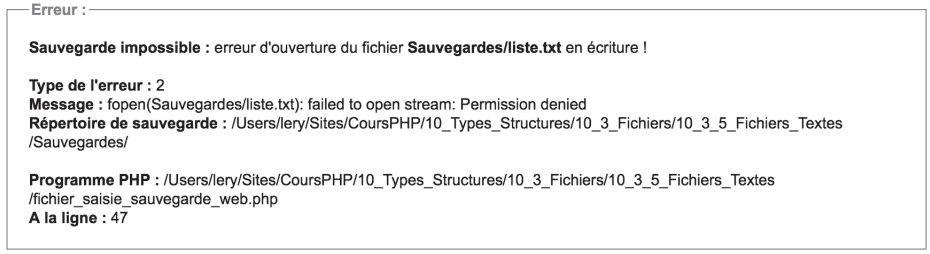


Figure 7.18 – **Message d'erreur si la sauvegarde est impossible.**

7.3 Faire un programme shell qui lit le fichier texte généré par le programme précédent. Chaque ligne contient les informations d'une personne : son Numéro, son Nom, son Prénom et son Age. Chaque personne lue sera conservée dans un tableau de personnes. Le tableau sera ensuite affiché.

Voici un exemple de fichier `liste_personnes.txt`.

```
0  Martin     Jean             34
1  De-La-Rue  Jean-Christophe  54
2  Dupont     Pierre           45
```

Voici un exemple d'exécution du programme attendu :

```
$ php fichier_lecture_personnes_shell.php
Nom du fichier : liste_personnes.txt
Fichier lu : Sauvegardes/liste_personnes.txt
-----------------------------------
ID Nom        Prénom           Age
-----------------------------------
0  Martin     Jean              34
1  De-La-Rue  Jean-Christophe   54
2  Dupont     Pierre            45
-----------------------------------
```

7.4 Adapter le programme précédent afin de saisir le nom du fichier dans un formulaire web (figure 7.19), puis d'afficher (figure 7.20) la liste des personnes qui ont été chargées dans un tableau de personnes.

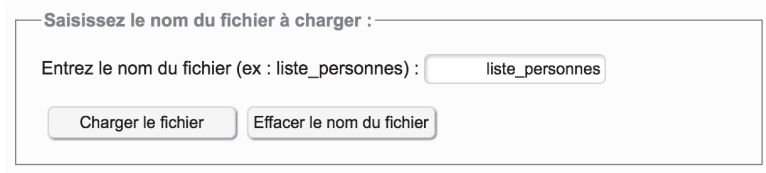


Figure 7.19 – **Lecture d'un fichier : formulaire de saisie.**

Message :

Chargement du fichier **Sauvegardes/liste_personnes.txt** :

Tableau des personnes lues à partir du fichier

ID	Nom	Prénom	Age
0	Martin	Jean	34
1	De-La-Rue	Jean-Christophe	54
2	Dupont	Pierre	45

Nombre de personnes lues : 3

Figure 7.20 – **Lecture d'un fichier : Affichage des personnes.**

Solutions

7.1 Cet exercice effectue la saisie d'une liste de personnes dans une interface web *via* les tableaux. Nous montrons comment effectuer l'équivalent d'une boucle de saisie à l'aide d'un formulaire HTML dans un programme PHP.

Algorithme du programme

L'architecture du programme est celle qui a été présentée à la figure 7.2. Un seul programme effectue les différents traitements, le formulaire de saisie pointe sur le programme lui-même. Il faut tester le contexte d'appel, et en particulier les variables de session qui indiquent quel bouton « action » a été sélectionné. Deux boutons du formulaire sont de type `"submit"` : « `Valider cette personne` » (`name="valider"`) et « `Terminer la saisie` » (`name="terminer"`).

Si l'exécution vient de « `Terminer la Saisie` » (test de `$_POST['terminer']`), on affiche le tableau de la liste des personnes s'il existe (`$tab_personnes` transmis comme variable de session), ou un message s'il n'existe pas.

Sinon l'exécution vient de « `Valider cette personne` » :

- On affiche le formulaire de saisie avec les champs vides.
- On récupère les données transmises *via* POST, de la précédente exécution.
- On traite les données (normalisation du nom, du prénom…).
- On affiche les informations après le formulaire si ce n'est pas la première exécution (variable de session `Afficher_Messages_champs` positionnée). Si tous les champs sont renseignés, on affiche la synthèse de la dernière personne saisie et on la range dans le tableau des personnes `$tab_personnes`, sinon on affiche la liste des champs vides.
- On positionne la variable de session `Afficher_Messages_champs` si ce n'est pas la première exécution.

Le programme

Voici le programme `tableau_saisie_affichage_web.php`. Le formulaire de saisie utilise une syntaxe HTML5 vérifiant les champs de saisie.

Listing 7.51 – Programme tableau_saisie_affichage_web.php

```
<?php
// Démarrage de la session AVANT d'écrire du code HTML afin de
// conserver l'information indiquant si c'est le premier accès
// et pour transmettre le tableau des personnes
session_start();
?>
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Saisie de plusieurs personnes</title>
    <!-- Feuille de style -->
    <link href="saisie_liste_personnes_1page.css" rel="stylesheet"
    type="text/css" />
 </head>
 <body>
   <?php
   define("WEB_EOL","<br/>");
   // L'appel provient d'un terminer, on affiche le tableau
   if (!empty($_POST['terminer']))
   {?>
    <table summary="Tableau des personnes">
     <caption>Tableau des personnes</caption>
     <thead>
     <tr> <!-- entête du tableau -->
        <th>ID</th>
        <th>Nom</th>
        <th>Pr&eacute;nom</th>
        <th>Age</th>
     </tr>
     </thead>
     <?php
     //Vérification que le tableau existe:au moins une saisie
     if (isset($_SESSION['tab_personnes']))
     {$tab_personnes=$_SESSION['tab_personnes'];
      foreach ($tab_personnes as $ID => $une_personne)
      {echo "<tr>";
       echo "<td>$ID</td>";
       foreach ($une_personne as $EtiqChamp => $ValChamp)
       {echo "<td>$ValChamp</td>";}
       echo "</tr>";
      }
     }
     else // Affichage d'un message dans le tableau
```

```
  { echo "<td colspan=\"4\"><b>Aucune donn&eacute;e &agrave;
  afficher</b></td>";
    echo "</tr>";
  }?>
 </table>
 <?php
}
// Affichage du formulaire et des informations en dessous
else
{ ?> <!-- --- Affichage du formulaire --- -->
  <form action="tableau_saisie_affichage_web.php" method="post">
   <fieldset>
   <legend>Saisissez les donn&eacute;es d'une nouvelle personne :
   </legend><br/>
   Entrez un nom (ex : Dupont) : <input type="text" name="Nom"
   size="20" maxlength="20" autofocus/><br/><br/>
   Entrez un pr&eacute;nom (ex : Jean) : <input type="text"
   name="Prenom" size="40" maxlength="40" /><br/><br/>
   Entrez un &acirc;ge (ex : 28) : <input type="text" name="Age"
   size="3" maxlength="3" pattern="[1-9][0-9]{1,2}" /><br/><br/>
   <input type="submit" name="valider" value="Valider cette
   personne" />
   <input type="reset" value="Effacer le formulaire" />
   <!-- Ajout du bouton pour terminer la saisie  -->
   <input type="submit" name="terminer" value="Terminer la Saisie" />
   </fieldset>
  </form>
  <?php
  // Affichage du résultat du traitement précédent
  // - soit le rangement dans le tableau $tab_personnes
  // - soit un message d'erreur (sauf premier affichage)
  // --- Récupération  des valeurs saisies ---
  if (isset($_POST['Nom'])) $Nom = $_POST['Nom'] ;
  else $Nom     = ''  ;
  if (isset($_POST['Prenom'])) $Prenom = $_POST['Prenom'] ;
  else $Prenom  = ''  ;
  if (isset($_POST['Age'])) $Age = $_POST['Age'] ;
  else $Age     = ''  ;
  // --- Traitement des données saisies ---
  // Suppression des espaces : début, fin et multiples
  // Remplace espace par moins : noms et prénoms composés
  // -- Nom --
  $Nom    = trim($Nom)                          ;
  $Nom    = str_replace('-', ' ', $Nom)         ;
  $Nom    = preg_replace('/\s{2,}/', ' ', $Nom) ;
  $Nom    = strtolower($Nom)                    ;
  $Nom    = ucwords($Nom)                       ;
  $Nom    = str_replace(' ', '-', $Nom)         ;
  // -- Prénom --
  $Prenom = trim($Prenom)                        ;
```

```
$Prenom = str_replace('-', ' ', $Prenom)        ;
$Prenom = preg_replace('/\s{2,}/', ' ', $Prenom);
$Prenom = strtolower($Prenom)                   ;
$Prenom = ucwords($Prenom)                      ;
$Prenom = str_replace(' ', '-', $Prenom)        ;
// -- Age --
$Age = trim($Age)                           ;
$Age = preg_replace('/\s{2,}/', ' ', $Age) ;
// Si ce n'est pas la première saisie
if (isset($_SESSION['Afficher_Messages_Champs']))
{// Vérification que la saisie n'est pas vide
 if (!empty($Nom) && !empty($Prenom) && !empty($Age) )
 {$Age = intval($Age)                          ;
  if (!(($Age>0)&&($Age<120)))
   echo "Le champ Age est invalide".WEB_EOL;
  else
  {?>
   <table summary="Rangement dans le tableau">
    <caption>Derni&eacute;re personne saisie</caption>
    <thead>
     <tr> <!-- entête du tableau -->
      <th>Nom</th>
      <th>Pr&eacute;nom</th>
      <th>Age</th>
     </tr>
    </thead>
    <tr>
     <td><?php echo $Nom    ; ?></td>
     <td><?php echo $Prenom ; ?></td>
     <td><?php echo $Age    ; ?></td>
    </tr>
   </table>
   <?php
   // Rangement dans $tab_personnes de la session
   $_SESSION['tab_personnes'][]=array($Nom,$Prenom,$Age);
   }
 }
 // Affichage messages d'erreur si les champs sont vides
 else
 {echo "<fieldset>";
  echo "<legend>Valeurs &agrave; renseigner :</legend><br/>";
  if (empty($Nom)) echo "Le champ Nom est vide".WEB_EOL;
  if (empty($Prenom)) echo "Le champ Pr&eacute;nom est vide".
  WEB_EOL;
  if (empty($Age)) echo "Le champ Age est vide".WEB_EOL;
  if ($Age==0) echo "Le champ Age est invalide".WEB_EOL;
  echo "</fieldset>";
 }
}
// Ce n'est pas la première saisie:la variable indiquant
```

```
      // d'afficher les messages d'erreurs est positionnée
      else
      {$_SESSION['Afficher_Messages_Champs']="oui";}
    } ?>
  </body>
 </html>
```

7.2 Cet exercice est une adaptation de l'exercice précédent et du programme `tableau_saisie_affichage_web.php`. Il ajoute la sauvegarde de la liste des personnes dans un fichier dont le nom est saisi dans un formulaire.

Algorithme du programme

L'architecture du programme est identique au programme précédent, seules la saisie du nom du fichier et la boucle de sauvegarde des personnes ont été ajoutées.

Un seul programme effectue les différents traitements, les formulaires de saisie pointent sur le programme lui-même. À chaque exécution de ce programme, il faut tester le contexte d'appel et en particulier les variables de session qui indiquent quel bouton « action » a été sélectionné. Trois boutons des formulaires sont de type `"submit"` : « `Valider cette personne` » (`name="valider"`), « `Terminer la saisie` » (`name="terminer"`) lors de la saisie des personnes, et « `Sauvegarder cette liste` » (`name="sauvegarder"`) lors de la saisie du nom du fichier.

Si l'exécution vient de la sélection du bouton « `Sauvegarder cette liste` » (test de `$_POST['sauvegarder']`) :

- on récupère et normalise le nom du fichier, puis on construit son nom complet à partir du répertoire courant et de l'extension « .txt » qui s'ajoute ou qui remplace une éventuelle extension saisie ;
- si l'ouverture échoue (`$f1=@fopen($NomF,"wt")`), on traite les erreurs ;
- sinon on sauvegarde le tableau `$tab_personnes` transmis comme variable de session, on confirme la sauvegarde, et on ferme le fichier.

Si l'appel vient de « `Terminer la Saisie` » (test de `$_POST['terminer']`) : le traitement est identique au programme précédent (affichage de `$tab_personnes` transmis en variable de session ou message d'erreur), et on ajoute un formulaire de saisie du nom du fichier de sauvegarde. Sinon l'exécution vient du bouton « `Valider cette personne` » : le traitement est identique au programme précédent.

Remarque

Le répertoire Sauvegardes doit exister comme sous-répertoire du répertoire du programme, et le propriétaire du serveur web (Apache) doit avoir le droit d'écrire dans ce répertoire.

Voici les commandes Unix à saisir, une seule fois, pour créer le sous-répertoire `Sauvegardes` dans le répertoire du programme PHP représenté par la syntaxe `<répertoire du programme php>`, et lui affecter les bons droits :

```
cd <répertoire_du_programme_php>
mkdir Sauvegardes
chmod a+w Sauvegardes
```

Le programme

Les lignes du programme `fichier_saisie_sauvegarde_web.php` identiques à `tableau_saisie_affichage_web.php` sont remplacées par des « … ». Seules la saisie du nom du fichier et la sauvegarde avec la gestion des erreurs sont détaillées.

Les erreurs produites par `fopen()`sont contrôlées grâce à l'opérateur « @ ». La fonction `error_get_last()`, retourne un tableau des éléments de la dernière erreur rencontrée. La fonction `realpath()`donne le chemin absolu du répertoire donné en argument soit le répertoire courant « . ».

Listing 7.52 – Programme fichier_saisie_sauvegarde_web.php

```
<?php
// Démarrage de la session AVANT d'écrire du code HTML
session_start();
?>
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Saisie de plusieurs personnes</title>
    <link href="chargement_sauvegarde_liste_personnes_1page.css"
    rel="stylesheet" type="text/css" />
 </head>
 <body>
   <?php
   define("WEB_EOL","<br/>");
   // L'appel provient d'un sauvegarder, on sauvegarde
   if (!empty($_POST['sauvegarder']))
   { // On récupère le nom du fichier
     if (isset($_POST['NomF'])) $NomF=$_POST['NomF'] ;
     else $NomF     = ''     ;
     if (empty($NomF))
     {?>
      <fieldset>
      <legend>Message :</legend><br/>
      <b>Sauvegarde impossible, le nom du fichier n'est pas indiqu&eacute;
      ! </b><br />
      </fieldset>
      <?php
     }
     else // On normalise le nom du fichier
```

```
{$NomF    = trim($NomF)                          ;
 $NomF    = preg_replace('/\s{2,}/', ' ', $NomF) ;
 $NomF    = strtolower($NomF)                    ;
 $NomF    = str_replace(' ', '_', $NomF)         ;
 // On prépare le nom du répertoire de sauvegarde
 $RepCourant=realpath(".");
 $RepSauv="Sauvegardes/";
 // On traite le nom du fichier
 $ElemChemin=pathinfo($NomF);
 $NomF=$RepSauv.$ElemChemin['filename'].".txt";
 // Ouverture du fichier en mode écriture
 // On empêche les messages d'erreurs du fopen avec @
 $f1 = @fopen($NomF, "wt");
 if (! $f1) // Erreur d'ouverture
 { // On affiche les messages d'erreur
  $tab_erreurs      = error_get_last()       ;
  $erreur_type      = $tab_erreurs['type']   ;
  $erreur_message   = $tab_erreurs['message'];
  $erreur_programme = $tab_erreurs['file']   ;
  $erreur_ligne     = $tab_erreurs['line']   ;
  ?>
  <fieldset>
  <legend>Erreur :</legend><br/>
  <b>Sauvegarde impossible :</b> erreur d'ouverture du fichier
  <b><?php echo $NomF; ?></b> en écriture  ! <br /><br/>
  <b>Type de l'erreur :</b> <?php echo $erreur_type ; ?><br />
  <b>Message :</b> <?php echo $erreur_message; ?><br />
  <b>Répertoire de sauvegarde :</b> <?php echo
  $RepCourant."/".$RepSauv; ?><br /><br />
  <b>Programme PHP :</b> <?php echo $erreur_programme; ?><br />
  <b>A la ligne :</b> <?php echo $erreur_ligne; ?><br />
  </fieldset>
  <?php
 }
 else // On affiche le résultat de la sauvegarde
 {?>
  <fieldset>
  <legend>Message :</legend><br/>
 <?php echo "Sauvegarde dans le fichier <b>$NomF</b> : "."<br />";?>
  </fieldset>
  <table summary="Tableau des personnes">
  <caption>Tableau des personnes sauvegard&eacute;es</caption>
  <thead>
  <tr>
  <!-- entête du tableau -->
   <th>ID</th>
   <th>Nom</th>
   <th>Pr&eacute;nom</th>
   <th>Age</th>
   </tr>
  </thead>
  <?php
  $nb_sauvegardes=0;
```

```
        $tab_personnes=$_SESSION['tab_personnes'];
        foreach ($tab_personnes as $ID => $une_personne)
        {echo "<tr>";
         echo "<td>$ID</td>";
         // -- Sauvegarde de l'ID ---
         fprintf($f1,"%d\t",$ID);
         foreach ($une_personne as $EtiqChamp => $ValChamp)
         {echo "<td>$ValChamp</td>";
          // -- Sauvegarde de chaque élément ---
          fprintf($f1,"%s\t",$ValChamp);
         }
         echo "</tr>";
         // -- Sauvegarde d'une fin de ligne ---
         fprintf($f1,"\n");
         $nb_sauvegardes++;
        }
        ?>
        </table>
        <?php
        // Fermeture du fichier de sauvegarde
        echo "<br />Nombre de personnes sauvegard&eacute;es :
        $nb_sauvegardes<br />";
        fclose($f1);
       }
      }
    }
    // L'appel provient d'un terminer, on affiche le tableau
    elseif (!empty($_POST['terminer']))
    {?>
     // Partie identique à tableau_saisie_affichage_web.php
     <table summary="Tableau des personnes">
     ...
     </table>
     <?php
     //Vérification que le tableau existe:au moins une saisie
     if (isset($_SESSION['tab_personnes']))
     {// On affiche saisit le nom du fichier de sauvegarde
      ?> <br/>
      <form action="fichier_saisie_sauvegarde_web.php" method="post">
      <fieldset>
      <legend>Saisissez le nom du fichier de sauvegarde :</legend><br/>
      Entrez le nom du fichier (ex : liste_personnes) : <input
     type="text" name="NomF" size="20" maxlength="20" autofocus/><br/><br/>
      <input type="submit" name="sauvegarder" value="Sauvegarder cette
      liste" />
      <input type="reset" value="Effacer le nom du fichier" />
      </fieldset>
      </form>
      <?php
     }
    }
    // Affichage du formulaire et des informations en dessous
    else
```

```
    { ?> <!-- --- Affichage du formulaire --- -->
      <form action="fichier_saisie_sauvegarde_web.php" method="post">
      <!-- Idem à tableau_saisie_affichage_web.php -->
      ...
      </form>
      <?php
      // Affichage du résultat du traitement précédent
      // Idem à tableau_saisie_affichage_web.php
      ...
    }
    ?>
  </body>
</html>
```

7.3 Le programme `fichier_lecture_personnes_shell.php` lit un fichier dont le nom est saisi au clavier et qui se trouve dans le répertoire `Sauvegardes`, conserve les éléments lus dans le tableau `$tab_personnes`, puis l'affiche. Aucun contrôle d'erreur sur le fichier n'est effectué.

Listing 7.53 – Programme fichier_lecture_personnes_shell.php

```
<?php
echo "Nom du fichier : ";
fscanf(STDIN,"%s",$NomF);
$NomF="Sauvegardes/".$NomF;
$f1=fopen($NomF, "r"); // Ouverture en lecture
echo "Fichier lu : $NomF".PHP_EOL;
// --- Boucle de chargement du fichier ---
while ($Tab_Info=fscanf($f1,"%d\t%s\t%s\t%d"))
{ list($ID,$nom,$prenom,$age) = $Tab_Info;
  $tab_personnes[$ID]=array($nom,$prenom,$age) ;
}
fclose($f1); // fermeture du fichier
// -- Affichage du tableau ---
echo "---------------------------------".PHP_EOL;
echo "ID\tNom\tPrénom\tAge".PHP_EOL;
echo "---------------------------------".PHP_EOL;
foreach($tab_personnes as $ID => $une_personne)
{list($nom,$prenom,$age) = $une_personne;
 echo "$ID\t$nom\t$prenom\t$age".PHP_EOL;
}
echo "---------------------------------".PHP_EOL;
?>
```

7.4 Le programme `fichier_affichage_chargement_web.php` est une adaptation web du programme précédent. Il lit le nom du fichier à charger dans un formulaire, puis affiche le tableau des personnes dans une autre page. Si l'exécution vient du bouton « `Charger le fichier` » (test de `$_POST['charger']`), on normalise le nom du fichier et on lui ajoute l'extension « .txt ». On charge le fichier dans le tableau `$tab_personnes` qui est défini comme variable de session, et on l'affiche. Sinon on affiche le formulaire de saisie du nom du fichier.

Remarque

Le répertoire Sauvegardes doit exister comme sous-répertoire du répertoire où se trouve le programme, et le propriétaire du serveur web (par exemple Apache) doit avoir le droit de lire le fichier et de se déplacer dans le répertoire Sauvegardes.

Voici les commandes Unix à saisir, une seule fois, pour affecter les bons droits au sous-répertoire `Sauvegardes` :

```
chmod a+x Sauvegardes
```

Pour lire le fichier existant, par exemple `liste_personnes.txt`, il faut s'assurer qu'il autorise sa lecture :

```
chmod a+r Sauvegardes/liste_personnes.txt
```

Dans le programme suivant, la ligne d'en-tête du formulaire :

```
<form action="fichier_affichage_chargement_web.php" method="post">
```

a été mise en commentaire HTML et remplacée par :

```
<form action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
```

Cette syntaxe utilise la variable super-globale `$_SERVER` (*cf.* chapitre 4).

Listing 7.54 – Programme fichier_affichage_chargement_web.php

```
<?php
// Démarrage de la session AVANT d'écrire du code HTML
session_start();
?>
<!DOCTYPE html>
<html>
 <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Chargement fichier de personnes</title>
     <link href="chargement_sauvegarde_liste_personnes_1page.css"
     rel="stylesheet" type="text/css" />
 </head>
 <body>
   <?php
   define("WEB_EOL","<br/>");
   // L'appel provient d'un charger, on charge le fichier
   if (!empty($_POST['charger']))
   { // On récupère le nom du fichier
     if (isset($_POST['NomFichier'])) $NomF = $_POST['NomFichier'] ;
     else $NomF = '' ;
     if (empty($NomF))
     {?>
      <fieldset>
      <legend>Message :</legend><br/>
      <b>Chargement impossible, le nom du fichier n'est pas indiqu&eacute;
```

```
 ! </b><br />
 </fieldset>
 <?php
}
else
{// On normalise le nom du fichier
 $NomF = trim($NomF)                        ;
 $NomF = preg_replace('/\s{2,}/', ' ', $NomF);
 $NomF = strtolower($NomF)                  ;
 $NomF = str_replace(' ', '_', $NomF)      ;
 // On prépare le nom du répertoire de Chargement
 $RepCourant=realpath(".")     ;
 $RepCharge="Sauvegardes/" ;
 // On traite le nom du fichier
 $ElemChemin=pathinfo($NomF);
 $NomF=$RepCharge.$ElemChemin['filename'].".txt";
 // Ouverture du fichier en mode lecture
 // On empêche l'affichage des messages d'erreurs avec @
 $f1 = @fopen($NomF, "rt");
 if (! $f1) // Erreur d'ouverture
 { // On affiche les messages d'erreur
  $tab_erreurs      = error_get_last()       ;
  $erreur_type      = $tab_erreurs['type']   ;
  $erreur_message   = $tab_erreurs['message'];
  $erreur_programme = $tab_erreurs['file']   ;
  $erreur_ligne     = $tab_erreurs['line']   ;
  ?>
  <fieldset>
  <legend>Erreur :</legend><br/>
  <b>Chargement impossible :</b> erreur d'ouverture du fichier
  <b><?php echo $NomF; ?></b> en lecture  ! <br /><br/>
  <b>Type de l'erreur :</b> <?php echo $erreur_type ; ?><br />
  <b>Message :</b> <?php echo $erreur_message; ?><br />
  <b>Répertoire de Chargement :</b> <?php echo
  $RepCourant."/".$RepCharge; ?><br /><br />
  <b>Programme PHP :</b> <?php echo $erreur_programme; ?><br />
  <b>A la ligne :</b> <?php echo $erreur_ligne; ?><br />
  </fieldset>
  <?php
 }
 else // On charge le tableau, et on l'affiche
 { ?>
  <fieldset>
  <legend>Message :</legend><br/>
  <?php echo "Chargement du fichier <b>$NomF</b> : "."<br />";?>
  </fieldset>
  <table summary="Tableau des personnes">
  <caption>Tableau des personnes lues à partir du fichier</caption>
  <thead>
  <tr>
```

```
        <!-- entête du tableau -->
         <th>ID</th>
         <th>Nom</th>
         <th>Pr&eacute;nom</th>
         <th>Age</th>
         </tr>
        </thead>
        <?php
        // Rangement dans $tab_personnes de la session
        $nb_Chargements=0;
        while ($Tab_Info=fscanf($f1,"%d\t%s\t%s\t%d"))
        { list($ID,$Nom,$Prenom,$Age) = $Tab_Info;
          $_SESSION['tab_personnes'][$ID]=array($Nom,$Prenom,$Age) ;
          $nb_Chargements++;
          echo "<tr>";
          echo "<td>$ID</td>";
          echo "<td>$Nom</td>";
          echo "<td>$Prenom</td>";
          echo "<td>$Age</td>";
          echo "</tr>";
        }
        ?></table><?php
        // Fermeture du fichier de Chargement
        echo "<br />Nombre de personnes lues : $nb_Chargements<br />";
        fclose($f1);
       }
      }
    }
    else // On affiche le formulaire de saisie
    { ?>
      <br/>
      <!-- <form action="fichier_affichage_chargement_web.php"
      method="post"> -->
      <form action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
       <fieldset>
       <legend>Saisissez le nom du fichier &agrave; charger :
       </legend><br/>
       Entrez le nom du fichier (ex : liste_personnes) : <input
       type="text" name="NomFichier" size="20" maxlength="20"
       autofocus/><br/><br/>
       <input type="submit" name="charger" value="Charger le fichier" />
       <input type="reset" value="Effacer le nom du fichier" />
       </fieldset>
      </form>
      <?php
    }?>
  </body>
 </html>
```