Technologies de l' Internet
PHP

Claude Guéganno

11 mars 2013

Table des matières

Chapitre 1

Bases du langage

1.1 Introduction

1.1.1 Historique

1994 : Rasmus Lerdof crée un ensemble de *scripts Perl* (`Personal Home Pages`) pour son utilisation personnelle.

1995 : mise à la disposition du public de `PHP`. `PHP3` exécute le code en le lisant.

2000 : `PHP4` compile le code et l'exécute.

2004 : `PHP5`

1.1.2 Caractéristiques

En bref :
- c'est un langage de `script` ;
- module intégré dans `Apache` ;
- écrit dans la page `html` : les codes `HTML`, `JavaScript` et `php` sont mélangés dans la page ;
- traité par le serveur (la page reçue ne contient plus de code `php`) ;
- résultat de requêtes à des `SGBD`, de calculs -> `HTML` dynamique : la page est fabriquée en fonction des résultats de calculs ou de requètes :
- gratuit ;
- syntaxe proche du `C` ;
- langage impératif, objet, réflexif, typage faible ;
- adresse officielle (beaucoup de documentations) `http://www.php.net` ;
- les fichiers contenant du code `php` finissent par *.php3* ou *.php* ;
- utilisation sous `Linux` ou `Windows` avec le serveur `Apache`.
- Intégrable également dans `IIS`

1.2 Outils de développement

Le développement d'applications `PHP` (sites ou logiciels) nécessite l'installation sur la machine de développement d'un serveur `HTTP` doté d'un module

PHP. Très souvent, la création d'un site dynamique ou d'un logiciel qui utilise cette technologie est associée à une base de données. PHP est par exellence un langage de script permettant le lien entre une base de données et l'internet.

1.2.1 Windows

EasyPHP Une solution simple pour démarrer rapidement le développement PHP sous *Windows* consiste à installer sur la machine de développement l'ensemble easyPhp qui intègre :

- Apache (serveur internet)
- PHP (langage de script)
- mySql (serveur de base de données)
- phpMyAdmin (site de gestion de la base de données).

EasyPHP est disponible gratuitement sur http://www.easyphp.org/.

WampServer Une variante pour *Windows* consiste à installer le logiciel WAMP http://www.wampserver.com/.

1.2.2 Linux

LAMP Concernant les systèmes *Linux*, il suffit d'installer l'ensemble appelé «LAMP» [1]. Par exemple, avec Ubuntu :

```
sudo tasksel install lamp-server
```

ou bien :

```
sudo apt-get install lamp-server^
```

1.2.3 Éditeur

À cela, on peut ajouter un éditeur facilitant l'écriture PHP. Il y en a beaucoup. Citons en particulier Komodo-Edit bien conçu pour l'édition PHP, javascript et HTML, qui intègre très bien la documentation PHP ainsi que la mise en correspondance des accolades, des parenthèses ...

Un autre critère important de choix est la fonction qui permet de rechercher, et de modifier des chaînes de caractères dans plusieurs fichiers. En effet, un site internet est composé d'un nombre de fichiers pouvant aller de quelques dizaines à plusieurs centaines.

Komodo est disponible sur http://www.activestate.com/komodo-edit.

1.3 PHP dans HTML

Le code PHP apparaît dans la page entre des balises :

Le '<' et le '>' sont des indicateurs de début et de fin de balise comme en html, pour les distinguer des balises html, trois approches sont autorisées :

1. Linux Apache Mysql Php

1. `<? code php ?>`
2. `<?php code php ?>`
3. `<SCRIPT language="php"> code php </SCRIPT>`

Les instructions dans les balises sont séparées par des `;`

Tout ce qui est entre `/*` et `*/` est un commentaire.

Exemple de fichier `php`, le fichier suivant est sur le serveur :

```
<HTML>
  <HEAD>
    <title>premiers pas en PHP</title>
  </HEAD>

  <BODY>
    <H1>Essai de script PHP</H1>

    <?php
       echo "Bonjour, il est ";
       echo date ("H:i:s");
    ?>
</BODY>
</HTML>
```

La page envoyée au navigateur est :

```
<HTML>
  <HEAD>
    <title>premiers pas en PHP</title>
  </HEAD>

  <BODY>
    <H1>Essai de script PHP</H1>
Bonjour, il est 17:05:38</BODY>
</HTML>
```

Remarque : on note que tout ce qui était entre les balises `<?` et `?>` a été interprété et transformé en code `HTML`.

1.4 Les données

1.4.1 Les variables

- La syntaxe pour déclarer une variable : `$ma_variable = expression;`
- attention à la casse (`$i` différent de`$I`) ;
- le nom de la variable commence avec une lettre ou un souligné et est constitué de lettres, chiffres, souligné ;
- pas de déclaration préalable ;
- en `PHP3` les variables sont assignées uniquement par valeurs.

Exemple

```
<?php
  $reel = 0.3;
  $entier = 22;
  $chaine = "World !";
  $phrase1 = "Hello $chaine!";
  $phrase2 = 'Hello $chaine$';

  print("un réel : $reel<BR>");
  print("un entier : $entier<BR>");
  print("une chaîne : $chaine<BR>");
  print("une phrase : $phrase1<BR>");
  print("du bon usage des guillemets : $phrase2");
?>
```

Données produites :

```
un réel : 0.3
un entier : 22
une chaîne : World !
une phrase : Hello World !!
du bon usage des guillemets : Hello $chaine$
```

Affichage des variables. Les lignes suivantes sont équivalentes :

```
echo "N = ".$N."<br>";
echo "N = $N<br>";
print("N =$N<br>");
```

Le caractère `$` est détecté par `PHP` et `$N` est remplacé par sa valeur. Si $N = 1234$, l'affichage produit par ces lignes est donc :

```
N = 1234
N = 1234
N = 1234
```

La substitution ne se fait pas si on utilise le caractère `'` pour délimiter la chaîne de caractères. Ainsi,

```
echo 'N = $N';
```

affiche

```
N = $N
```

– En `PHP4` les variables peuvent être assignées par références (elles deviennent un alias sur la variable qu'elles référencent) :

```
<?php
$chaine1 = "Bonjour";
$chaine2 = &$chaine1;   // Reference $chaine1 par $chaine2.
$chaine2 = "$chaine2 tout le monde";
print("$chaine1\n$chaine2");
?>
```

donne :

```
Bonjour tout le monde
Bonjour tout le monde
```

– type donné automatiquement (`integer`, `double`, `string`, `array`, `object`).

– pour écrire un guillemet dans une chaine, utiliser \".

```
<?php
  $chaine = "<a href=\"index.html\">accueil</a>";
  echo $chaine;
?>
```

donne :

```
<a href="index.html">accueil</a>
```

Fonctions de conversion

1. `intval`, `strval`, `doubleval`
   Exemple : `$entier = intval($reel);`
2. `(integer)`, `(string)`, `(double)`
   Exemple : `$entier = (integer)$reel;`
3. `int settype(var, type)` : force la variable *var* à prendre le type *type* et renvoie `0` en cas d'échec.
   Exemple : `settype($i,"integer");`
   Les type possibles sont `"integer"`, `"double"`, `"string"`, `"array"` et `"object"`.
4. `strval(argument)` retourne l'argument sous forme de chaîne.

Fonctions pour connaître le type

– `gettype($v)` : renvoie une chaîne de caractère donnant le type de `$v`.
– `is_array($v)`, `is_double($v)`, `is_integer($v)`, `is_object($v)`, `is_string($v)` donnent 1 si `$v` est du type précisé et 0 sinon.

Variables d'environnement

Il existe des variables prédéfinies, la plupart dépendent du serveur qui appelle le script `php`.

Pour obtenir la liste complète il suffit d'exécuter le *script* ci-dessous, une page sera affichée contenant des tableaux qui décrivent la configuration, les variables d'environnement `Apache` sont dans le tableau `«Apache Environment»`.

```
<?php
   phpinfo();
?>
```

Quelques unes des variables d'environnement (`Apache`)

`HTTP_USER_AGENT`	navigateur du client
`REQUEST_METHOD`	méthode utilisée pour appeler la page (`GET`, `HEAD`, `POST`, `PUT`) ;

Variables PHP

`HTTP_GET_VARS`	tableau associatif pour la methode get (nom, valeur)
`HTTP_POST_VARS`	tableau associatif pour la methode `post` (nom, valeur)
`HTTP_COOKIE_VARS`	tableau associatif pour la methode `cookie` (nom du `cookie`, contenu)
`GLOBALS`	tableau associatif qui contient les variables globales du *script*, la clé est le nom de la variable (sans le dollar) et la valeur est le contenu de la variable.

Variables dynamiques

Il est possible de définir un nom de variable déclaré et utilisé dynamiquement. Une variable dynamique prend comme nom la valeur d'une autre variable.

- Affectation : `$$nom_var = valeur;`
- Lecture de la valeur : `$$nom_var`

```
<?php
  $ch = "bonjour";
  $$ch = "tout le monde"; // '$bonjour' devient aussi une variable
  echo "$ch<BR>";
  echo "${$ch}<BR>";
  echo "$bonjour<BR>";
?>
```

donne :

```
bonjour
tout le monde
tout le monde
```

1.4.2 Les constantes

Définition d'une constante

Une constante est définie par l'instruction `define` :

```
define ("NOM_CONST", valeur);
```

On y accède directement par son nom (Ex : `NOM_CONST`. On peut vérifier qu'une constante est définie avec le prédicat `defined` :

```
if ( defined(NOM_CONST) )  ...
```

Exemple

```
<?php
   define("SITE", "http://claude.gueganno.free.fr");
   print("à visiter: ".SITE);
?>
```

Constantes prédéfinies

`TRUE`	vrai
`FALSE`	faux
`__FILE__`	nom du fichier du *script* interprété
`__LINE__`	numéro de la ligne du script
`PHP_VERSION`	version de PHP
`PHP_OS`	système sur lequel PHP exécute le script

```
<?php
  echo PHP_VERSION."<BR>";
  echo PHP_OS."<BR>";
  echo __LINE__."<BR>";
  echo __FILE__."<BR>";
?>
```

1.4.3 Les tableaux

Tableaux à une dimension

Il y a trois possibilités équivalentes pour initialiser un tableau :

```
$tab = array("b","o","n");
$tab[0] = "b"; $tab[1] = "o"; $tab[2] = "n";
$tab[] = "b"; $tab[] = "o"; $tab[] = "n";
```

(pas de déclaration préalable).

L'accès à un élément dans tableau de n cases se fait :

```
$tab[$indice]
```

avec `$indice` $\in [0, n-1]$.

Nombre d'éléments dans un tableau : avec la fonction `count`

```
$nb = count($tab);
```

Exemple :

```
<?php
  echo "essai avec array : ";
  $tab = array("b", "o", "n");
  $taille = count($tab);
  for($i=0; $i<$taille; $i++)
      echo $tab[$i];

  echo "<BR>essai avec [0], [1], ... : ";
  $tab2[0] = "h";  $tab2[1] = "e";  $tab2[2] = "l";
  $tab2[3] = "l";  $tab2[4] = "o";
  $taille = count($tab2);
  for($i=0; $i<$taille; $i++)
      echo $tab2[$i];

  echo "<BR>essai de [] : ";
  $tab[] = "j";  $tab[] = "o";  $tab[] = "u";  $tab[] = "r";
```

```
  $taille = count($tab);
  for($i=0; $i<$taille; $i++)
     echo $tab[$i];
?>
```

Sortie du programme :

```
essai avec array : bon
essai avec [0], [1], ... : hello
essai de [] : bonjour
```

Tableaux associatifs

L'accès aux données se fait par un nom (une clé) et non plus par un index.

Initialisations :

1. par éléments :

   ```
   $nomtab["cle"] = valeur;
   ```

2. tout (ou partie) du tableau :

   ```
   $nomtab = array("cle1"=>valeur1,"cle2"=>valeur2,...);
   ```

L'accès à un élément se fait par la clé qui lui est associé :

```
$nomtab["cle"];
```

Exemples de parcours

```
/* avec une boucle */
reset($nomtab);
while ($cle = key($nomtab)){
   $val = pos($nomtab);
   print("$cle = $val<BR>\n");
   next($nomtab);
}

reset($nomtab);
while (list($cle, $valeur) = each($nomtab)){
   echo "<BR>$cle = $valeur";
}
```

Tableaux de dimension ≥ 2

```
$saison = array(
  "ete"=>array("juin","juillet","aout","septembre"),
  "automne"=>array("septembre","octobre","novembre","decembre"),
  "hiver"=>array("decembre","janvier","fevrier","mars"),
  "printemps"=>array("mars","avril","mai","juin"));

print($saison["printemps"][2]);
```

affiche `mai`.

Fonctions de manipulation des tableaux

`int count(tab)`	nombre d'éléments
`int sizeof(tab)`	nombre d'éléments
`array explode(sep, chaine)`	transforme une chaine en tableau en fonction du séparateur.
`string implode(tab, sep)`	opération inverse
`val max(tab)`	retourne la plus grande valeur d'un tableau
`val min(tab)`	retourne la plus petite valeur du tableau
`shuffle(tab)`	change aléatoirement l'ordre des éléments
`array_walk(tab,'fonc')`	applique la fonction `fonc` au tableau `tab`

Itérateurs

PHP propose quelques fonctions de parcours d'un tableau. Chaque élément du tableau est en fait une paire $\{cl, valeur\}$. Les opérations portent soit sur la valeur, soit sur la clé.

Exemple

```
$T = array( "A", "B", "C", "D", "E");
```

Les clés sont attribuées automatiquement (entiers de 0 à 4).

`current(tab)`	valeur de l'élément courant ; `pos(tab)` fait la même chose.
`tableau each(tab)`	clé et valeur pour l'endroit où est l'itérateur (fait avancer de 1 l'itérateur)
`valeur end(tab)`	place l'itérateur sur le dernier élément du tableau
`clé key(tab)`	index de l'élément courant
`valeur next(tab)`	avance l'itérateur d'une case, retourne sa valeur
`valeur prev(tab)`	recule d'une case et retourne sa valeur
`valeur reset(tab)`	itérateur sur le premier élément du tableau

Exemple

```
<html>
<head>
<title> Tableaux en PHP </title>
</head>

<body>
 <h1>Tableaux</h1>
<?php
$T = array( "A", "B", "C", "D", "E");
echo implode($T,"; ");
echo "<br>";
echo "current(\$T) --> ".current($T)."<br>";
```

```
$T2 = each($T);
echo "each(\$T) --> [".$T2[0]." , ".$T2[1]."]<br>";
echo "current(\$T) --> ".current($T)."<br>";
echo "key(\$T) --> ".key($T)."<br>";
echo "next(\$T) --> ".next($T)."<br>";
echo "end(\$T) --> ".end($T)."<br>";
echo "prev(\$T) --> ".prev($T)."<br>";
echo "current(\$T) --> ".current($T)."<br><br>";
?>
</body>
</html>
```

FIGURE 1.1 – *Fonctions de parcours de tableau.*

Les tris

On peut les séparer en deux catégories :

1. ceux qui se font sans modification des clés (`arsort`, `ksort`, `asort`, `uasort`)
2. ceux qui effacent les clés.

`arsort(tab)`	trie par ordre décroissant, l'index associé se déplace avec les valeurs
`asort(tab)`	trie par ordre croissant, l'index associé se déplace avec les valeurs
`ksort(tab)`	trie le tableau par ses clés
`ksort(tab)`	trie le tableau en ordre inverse par ses clés
`rsort(tab)`	trie par ordre décroissant, efface les valeurs des clés
`sort(tab)`	trie par ordre croissant, efface la valeur des clés
`uasort(tab, comp)`	trie selon la fonction `comp` (définie par l'utilisateur), conserve la valeur des clés
`usort(tab, comp)`	trie selon la fonction, efface la valeur des clés
`uksort(tab, comp)`	trie les clés selon la fonction `comp` définie par l'utilisateur

```
<html>
<head>
<title> Tableaux en PHP </title>
</head>

<body>
 <h1>Tableaux</h1>

<?php
```

```
/* Affiche les couples [clé, valeur] tu tableau $tab */
function affiche($tab) {
    $n = sizeof($tab);
    reset($tab);
    for ($i=0; $i<$n; $i++) {
        $t = each($tab);
        echo "[".$t[0].",".$t[1]."]; ";
    }
}


$T1 = array( "D", "C", "E", "B", "A");
$T2=$T1;
echo "<b>Valeur initiale du tableau avant chaque appel:</b><br>\n";
echo "\$T2 = ".implode($T2,"; ")."<br>\n";
affiche($T2);
echo "<hr>";

arsort($T2);
echo "arsort(\$T2) --> "; affiche($T2); echo "<br>\n";

$T2 = $T1; asort($T2);
echo "asort(\$T2) --> "; affiche($T2); echo "<br>\n";

$T2 = $T1;   ksort($T2);
echo "ksort(\$T2) --> "; affiche($T2); echo "<br>\n";

$T2 = $T1; sort($T2);
echo "sort(\$T2) --> "; affiche($T2); echo "<br>\n";

$T2 = $T1; rsort($T2);
echo "rsort(\$T2) --> "; affiche($T2); echo "<br>\n";

echo "<hr>\n";
?>
</body>
</html>
```

FIGURE 1.2 – *Fonctions de tri de tableau.*

Fonctions liées à la notion d'ensemble

`tableau array_count_values (tab)`	retourne un tableau associatif qui donne en clé les valeurs du tableau et en valeur le nombre d'apparitions
`tableau array_diff (tab1, tab2, ...)`	retourne un tableau contenant toutes les valeurs de `tab1` qui ne sont pas dans `tab2, ...`
`tableau array_keys (tab)`	retourne un tableau contenant toutes les clés de `tab`.
`tableau array_values (tab)`	retourne un tableau contenant toutes les valeurs de `tab`.
`bool in_array (val, tab)`	retourne *vrai* si la valeur `val` est dans le tableau `tab`

Remarque : `tableau array_count_values()` et `tableau array_diff()` à partir de PHP4.

Gestion d'un tableau en pile ou en file

`valeur array_pop (tab)`	retire et retourne le dernier élément du tableau
`array_push (tab, e1, e2, ...)`	empile les éléments `e1`, `e2`, ... à la fin du tableau `tab`
`array_unshift(tab, e1, e2, ...)`	insère les éléments `e1`, `e2`, ... au début du tableau `tab`
`array_shift(tab)`	retire et retourne le premier élément du tableau `tab`

1.5 Les classes

Une classe est une collection de variables (ou attributs) et de fonctions (ou propriétés).

Définition d'une classe. Cela revient à définir un nouveau type. Syntaxe générale :

```
class nomclasse{
   /* attributs */
   var $nomvar_1;
   ...
   var $nomvar_n;

   /* constructeur */
   function nomclasse($arg1=defaut1, $arg2=defaut2, ...){
      instructions;
   }

   /* propriétés */
   function nomfonc_1(){
     instructions;
     return x;
   }
   ...
   function nomfonc_n($arg1=defaut11, $arg2, ...){
     instructions;
   }
}
```

Le constructeur est facultatif. Il permet d'initialiser les attributs de la classe. Il ne peut pas être surchargé, mais on peut préciser des arguments «par défaut», ce qui revient finalement au même.

Création d'une instance de la classe : elle se fait avec l'opérateur `new`.

```
objet = new nomclasse(...);
```

Si le constructeur reçoit des arguments, ils doivent être obligatoirement passés. S'il n'y a pas de constructeur, ou si celui–ci ne reçoit pas d'argument :

```
objet = new nomclasse;
```

Héritage : il se fait avec le mot-clé `extends`. Le constructeur de la classe parent n'est pas appelé systématiquement. Cependant, si la classe dérivée n'a pas de constructeur, c'est celui de la classe parent qui est appelée.

Exemple

```
<html>
<head>
<title> Classes </title>
</head>
```

```
<body>
 <h1>Classes</h1>

<?php
/* --------- classe pile ----------------- */
class pile {
    /* attributs */
    var $N;
    var $tab;
    var $nom;

    /* constructeur */
    function pile($nom="P"){
        $this->N=0;
        $this->nom = $nom;
        echo "Création d'une pile<br>";
    }

    /* propriétés */
    function ajoute($x){
        $this->tab[$this->N] = $x;
        $this->N++;
    }

    function retire() {
        $this->N--;
        return $this->tab[$this->N];
    }
}

/* --------- classe pile2 ---------------- */
class pile2 extends pile {

    function affiche() {
        echo $this->nom." --> ";
        for ($i=0; $i<$this->N; $i++)
            echo $this->tab[$i]."; ";
    }
}

/* -------- programme PHP ---------------- */
$P = new pile2("cours PHP");
$P = new pile2;
$P->ajoute("php");
$P->ajoute("java");
$P->affiche(); echo "<br>";
echo $P->retire()."<br>";
$P->affiche(); echo "<br>";
?>

</body>
```

```
</html>
```

FIGURE 1.3 – *Exemple sur les classes.*

1.6 Expressions et instructions

1.6.1 Les opérateurs

Opérateurs arithmétiques

`+`	addition	binaires
`-`	soustraction	|
`*`	multiplication	|
`/`	division	|
`%`	modulo	|
`++`	incrémentation	unaires
`--`	décrémentation	|

Opérateurs logiques

`&& and`	et	binaires
`|| or`	ou	|
`! not`	négation	unaire

Opérateurs sur les *bits*

`&`	et `(1 & 1)` $\rightarrow 1$
`|`	ou `(1 | 0)` $\rightarrow 1$
`^`	ou exclusif `(1 ^ 1)` $\rightarrow 0$
`~`	non

Opérateurs d'affectation

signe	utilisation	équivalent
`=`	`$x = $y`	`$x = $y`
`+=`	`$x += $y`	`$x = $x + $y`
`-=`	`$x -= $y`	`$x = $x - $y`
`*=`	`$x *= $y`	`$x = $x * $y`
`/=`	`$x /= $y`	`$x = $x / $y`
`%=`	`$x %= $y`	`$x = $x % $y`
`++`	`$x++`	`$x = $x + 1`
`--`	`$x--`	`$x = $x - 1`

Opérateurs relationnels

`$x == $y`	vrai si `$x` = `$y`
`$x != $y`	vrai si `$x` différent de `$y`
`$x <= $y`	vrai si `$x` inférieur ou égal à `$y`
`$x >= $y`	vrai si `$x` supérieur ou égal à `$y`
`$x > $y`	vrai si `$x` supérieur à `$y`
`$x < $y`	vrai si `$x` inférieur à `$y`
`$x === $y`	PHP4 : vrai si `$x` = `$y` et qu'ils sont du même type
`$x !== $y`	PHP4 : vrai si `$x` différent de `$y` ou qu'ils n'ont pas le même type

1.6.2 Les structures de commande

Convention pour les valeurs booléennes

- `false` → 0
- `true` → tout le reste.

Structure conditionnelle `if`

Si l'expression est vraie exécuter l'instruction ou les instructions dans le bloc.

```
if (expression){
  instruction1;
  ...
  instructionN;
}
```

Les accolades sont facultatives lorsque le bloc ne contient qu'une seule instruction :

```
if (expression)  instruction1;
```

Cette remarque reste vraie pour toutes les structures qui suivent.

Structure conditionnelle `if ...else`

Si l'expression est vraie exécuter les instructions du bloc 1 sinon exécuter celles du bloc 2.

```
if (expression){
  instruction1;
  ...
}
else {
  instruction1;
  ...
}
```

Cette structure peut être remplacée par l'opérateur ternaire `?:`

```
condition ? instruction si true : instruction si false;
```

Structure conditionnelle `if ...elseif ...else`

Si l'expression est vraie exécuter le bloc 1, sinon, si l'expression 2 est vraie exécuter le bloc 2, sinon si ..., sinon exécuter le dernier bloc.

```
if (expression){
  instruction1;
  ...
}
elseif (expression2){
  instruction2;
  ...
}
elseif (expression3){
```

```
  instruction3;
  ...
}
else {
  instructionN
  ...
}
```

Structure conditionnelle `switch`

Selon l'expression exécuter des instructions.

```
switch (expression){
   case val1 : instructions_1; break;
   case val2 : instructions_2; break;
   ...
   default : instructions_N;
}
```

Les valeurs ne peuvent pas être des tableaux ou des objets.

Structure répétitive `while`

Tant que l'expression est vraie faire les instructions

```
while (expression) {
   instructions;
}
```

Structure répétitive `do ... while`

Faire les instructions tant que l'expression est vraie

```
do{
  instructions;
}
while (expression);
```

Structure répétitive `for`

C'est une variante du `while`

```
for (instruction1; expression; instruction2){
   instructions;
}
```

Cette écriture équivaut à :

```
instruction1;
while (expression) {
   instructions;
   instruction2;
}
```

Structure répétitive `foreach`

Permet d'appliquer un bloc d'instructions *pour chaque* élément d'un tableau (`PHP4` uniquement).

```
foreach($tab as $value){
   instructions;
}
```

Pour tout le tableau `$tab`, à chaque itération la valeur de l'élément courant est assignée à la variable `$value` et le pointeur sur le tableau est avancé d'une case. Au départ le pointeur sur le tableau est automatiquement mis sur la première case par le `foreach`.

Exemple :

```
$tab = array(1,2,3,4);
foreach($tab as $v){
   $v++;
}
```

Interruption de boucles

- `break` interrompt les boucles `for`, `while`.
- `continue` interrompt l'exécution d'une itération et reprend à l'itération suivante.
- `exit` interrompt le script

1.6.3 Les fonctions

1.6.4 Définition d'une fonction

Il faut utiliser le mot–clé `function` :

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
   instructions;
   return $retour;
}
```

Une fonction doit être définie avant d'être utilisée.

Passage des arguments

```
function foo ($arg_1, &$arg_2, $arg3="defaut", ... ) {
   ...
}
```

dans cet exemple,

- `$arg1` est passé par valeur ;
- `$arg2` est passé par référence ; si sa valeur est modifiée dans la fonction, elle est modifiée pour tout le programme ;
- `$arg2` est optionnel, il peut ne pas être passé lors de l'appel de la fonction puisqu'il a une valeur par défaut.

Exemple Le fichier suivant contient un fonction chargée de dessiner un échiquier carré dont le nombre de lignes est passé en argument. La valeur par défaut de cet argument est 4. Le rendu de la page est donné par la figure 1.4.

```
<html>
<head>
<title> Fonctions PHP </title>
</head>

<body>
 <h1>Fonctions</h1>

<?php
function damier($n=4){
    echo "<table border=\"1\">\n";
    $couleur1 = 0;
    $i = 0;
    while ($i<$n) {
        $couleur1 = 1 - $couleur1;
        $couleur2 = $couleur1;
        echo "<tr>";
        for ($j=0; $j<$n; $j++) {
            echo "<td bgcolor=\"".($couleur2?"white":"black")."\">";
            echo "   </td>";
            $couleur2 = 1 - $couleur2;
        }
        echo "</tr>\n";
        $i++;
    }
    echo "</table>";
}

echo "<center>\n";
damier();
echo "<br>";
damier(7);
echo "</center>\n";
?>

</body>
</html>
```

FIGURE 1.4 – *Exemple de fonction.*

1.6.5 Inclusion de fichiers

Inclusion avec `include`

L'instruction `include` réalise l'inclusion et l'évaluation d'un fichier. Le fichier est réévalué à chaque nouvelle inclusion.

Exemple :

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
   include $files[$i];
}
```

Attention à l'utilisation conditionnelle de cette instruction : puisqu'elle est remplacée par les instructions du fichier, il faut qu'elle figure dans un bloc d'instructions (marqué par des accolades).

```
/* Mauvaise écriture */
if ($condition) include($fichier);
else include($autreFichier);

/* Bonne écriture */
if ($condition) { include($fichier); }
else { include($autreFichier);}
```

Inclusion avec `require`

Cette instruction est semblable à `include`, à ceci près que le code est inséré même s'il n'est pas exécuté. Il faut donc éviter d'utiliser le `require` dans une structure conditionnelle.

1.7 Envoi de données vers un script

1.7.1 Utilisation de la fenêtre «URL» du navigateur

Il suffit de compléter l'URL avec la liste des paramètres séparés par le caractère `&`. Un `?` sépare le nom du fichier de la liste de paramètres.

Exemple :

```
http://localhost/html/recepdata.php?N=6&nom=Zaza&X=3.14
```

1.7.2 Appel à partir d'un lien HTML

La même méthode fonctionne en général. Il est cependant conseillé de remplacer le `&` par son expression HTML `&`.

Exemple

```
<html>
<head><title>Envoi de données</title></head>
<body>
 <h3>Envoi de données</h3>
 <a href="recdata.php?N=99&amp;nom=Zaza&amp;X=3.1415">
    Cliquer ici!
 </a>
</body>
</html>
```

1.7.3 Récupération des données

Lorsque les données sont passées par l'URL, elles se récupèrent en PHP dans le tableau global associatif `$_GET[]`. Les index sont les noms des paramètres.

Exemple Voici le script de réception de l'URL `recdata.php?N=99&nom=Zaza&X=3.1415`

```
<html>
<head>
<title>sendData</title>
</head>
<body bgcolor="#FFFFFF">
<ol>
<?php
     echo "<li>Paramètre N : ".$_GET["N"];
     echo "<li>Paramètre Nom : ".$_GET["nom"];
     echo "<li>Paramètre X : ".$_GET["X"];
?>
</ol>
</body>
</html>
```

Lorsqu'on clique sur le lien de l'exemple précédent (ou qu'on rentre directement le lien dans le champ URL du navigateur), il y a exécution du script (figure 1.5).

FIGURE 1.5 – *Envoi et réception de données*

1.8 Données d'un formulaire

Les données d'un formulaire sont directement accessibles à partir du nom affecté à chaque variable dans le fichier `HTML`. Ce nom sert d'index dans le tableau `$_GET[]`, ou bien dans le tableau `$_POST[]`, selon la méthode choisie pour poster le formulaire.

Exemple : le formulaire suivant met en œuvre les principaux éléments.

```
<table border=1><tr><td> <!-- Cadre pour le formulaire -->
<form ENCTYPE="multipart/form-data"  action="simpleForm.php" method="post">
     <b>Menu</b> <select name="civilite" >
                    <option  selected>Monsieur</option>
                    <option>Madame</option>
                    <option>Mademoiselle</option>
                 </select>
     Nom : <input type="text" size="16" maxlength="32" name="nom" value="">
     Prenom : <input type="text" size="16" maxlength="32" name="prenom" value="">
     <br />
     Code postal:<input type="text" size="5" maxlength="5" name="codepostal" value="75">
        
     Ville : <input type="text" size="16" maxlength="16" name="ville" value="PARIS">
     <br /> <br /> <hr />
     <table border=1><tr>
     <td> Centres d'intérêts :
       <select name="centre[]"  size=3 multiple>
         <option selected value="mus">Musique</option>
         <option selected value="lit">Littérature</option>
         <option   value="spo">Sport</option>
         <option   value="tec">Technique</option>
         <option   value="aut">Autre</option>
       </select>
       <input type="hidden" name="N" value="5"></td>
       <td><table border="0">
         <tr><td> Langue parlée</td></tr>
         <tr><td>Français</td><td>Anglais</td><td>Autre</td></tr>
         <tr>
          <td><input type="checkbox" checked name="franc"></td>
          <td><input type="checkbox" checked name="ang"></td>
          <td><input type="checkbox" name="autre"></td>
         </tr>
       </table>
     </td>
     <td align="center">Contact par:<br />
```

```
      Mail <input type="radio" name="exp" checked value="mail">
        ou courrier <input type="radio" name="exp"  value="courier">
    </td></tr>
    </table>
    <hr>
    <center> Mot de passe:
    <input type="password" size="8" maxlength="8" name="mdp1" value="">
          
       Confirmer le mot de passe
    <input type="password" size="8" maxlength="8" name="mdp2" value="">
    <hr />
    Remarques:<br>
    <textarea name="remarque" rows="3" cols="50"  wrap="virtual"> Ecrire ici!
    </textarea>
    <hr /> <input type="reset" value="RAZ">
                    
           <input type="submit" value="Valider">
    </center>

</form>
</td></tr></table>
```

Le rendu de la page est donné par la figure 1.6.

FIGURE 1.6 – *Formulaire*

Les données du formulaire sont postées vers le fichier de traitement *simpleForm.php*. Le traitement consiste simplement ici à afficher les variables reçues sur le serveur.

```
<?php
   echo "civilité: ".$_POST["civilite"]."<br />".
        "nom: ".$_POST["nom"]."<br />".
        "prenom: ".$_POST["prenom"]."<br />".
        "CP: ".$_POST["codepostal"]."<br />".
        "Ville: ".$_POST["ville"]."<br />".
        "Centres: ".$_POST["centre"]."<br />".
        "Hidden: ".$_POST["N"]."<br />".
        "Français: ".$_POST["franc"]."<br />".
        "Anglais: ".$_POST["ang"]."<br />".
        "Autre: ".$_POST["autre"]."<br />".
        "Exp: ".$_POST["exp"]."<br />".
        "MDP1: ".$_POST["mdp1"]."<br />".
        "MDP2: ".$_POST["mdp2"]."<br />".
        "Rem: ".$_POST["remarque"]."<br />";
?>
<h1>Tableaux</h1>
<?php
   echo "centre: ";
   print_r ($_POST["centre"]);
?>
```

L'exécution du script produit le résultat de la figure 1.7.

FIGURE 1.7 – *Traitement du formulaire*

Remarques :

- Le traitement du forulaire peut également se faire en récupérant les valeurs des paramètres dans le tableau `$_GET` : avec `GET`, les noms et les valeurs des paramètres apparaissent dans la barre d'URL.
- Pour les champs de texte (`<input type="text" name="nom" ...>`) la valeur du paramètre est directement récupérée dans la variable `$_POST["nom"]`.
- Pour les sélections multiples (Ex. `<select name="centre[]" size=3 multiple>`), le paramètre donné dans le fichier HTML est un tableau. Les éléments sélectionnés prennent place dans ce tableau. Le nombre d'éléments du tableau correspond au nombre d'éléments sélectionnés dans le formulaire. Ils sont indexés par des entiers, en partant de 0.
- Pour les sélections simples (un seul choix possible, Ex : `<select name="civilite" >`), le formulaire envoie une variable simple (et non pas un tableau) qui prend la valeur de l'option sélectionnée.
- Les champs de type `checkbox` ne sont envoyés que s'ils sont cochés. Dans ce cas la valeur est `"on"`. Il faut donc, pour ces champs, tester leur existence avec la fonction `isset()` qui renvoie `true` si une variable existe. Ainsi, pour traiter convenablement ces champs il faudrait écrire :

```
if (isset($_POST["autre"])) { // la variable existe ...
```

1.9 Envoi de fichier au serveur

L'envoi d'un fichier au serveur fait l'objet d'un traitement particulier. On considère, par exemple, le formulaire suivant, qui permet d'envoyer un fichier quelconque vers un serveur (Fig 1.8) :

```
<html>
<head>
<title>envoi de fichier au serveur</title>
</head>
<body>
<form enctype="multipart/form-data"
      action = "envF.php"
      method = "POST">
   <input type="hidden" name="MAX_FILE_SIZE" value="100000">
   <input name="fp" type="file" size="100"><br>
   <input type="submit" value="<<Envoyer>>">
</form>
</body>
</html>
```

Le champ dédié pour un envoi de fichier est un `input` de type `file`. Dans la balise `<form>`, il faut préciser `enctype="multipart/form-data"`.

FIGURE 1.8 – *Envoi de fichier vers le serveur*

Réception du fichier . Le script chargé de traiter le fichier récupère les informations sur le fichier dans le tableau associatif `$_FILES["fp"]`, `"fp"` étant dans notre exemple le nom donné au champ de saisie du nom de fichier.

L'exemple de script suivant réalise l'affichage des informations recueillies sur le fichier reçu, puis stocke le fichier avec son nom d'origine dans un répertoire appelé `stock`.

```
<html>
<head>
<title>Réception de fichier</title>
</head>
<body>

<?php

$tmp_name = $_FILES['fp']['tmp_name'];
$name = $_FILES['fp']['name'];
$size = $_FILES['fp']['size'];
$type = $_FILES['fp']['type'];
$erreur = $_FILES['fp']['error'];

echo "Nom temporaire : ".$tmp_name."<br />".
     "Nom : ".$name."<br />".
     "Taille : ".$size."<br />".
     "Type : ".$type."<br />".
     "Erreur : ".$erreur."<br />";
// Ici: vérifier le type du fichier ...

// sauvegarde dans un répertoire 'stock'
move_uploaded_file($tmp_name,"stock/$name");

?>

</body>
</html>
```

Le fait de stocker le fichier reçu dans un fichier temporaire sécurise le site. Le programme PHP peut renommer le fichier conformément au type d'information attendue. Ceci interdit à un utilisateur de charger un fichier PHP vers le site, puis de l'exécuter.

L'exécution du script donné en exemple est rapporté par la figure 1.9.

FIGURE 1.9 – *Réception d'un fichier*

1.10 Les `cookies`

Les *cookies* sont de petites informations à durée de vie limitée qu'un serveur peut écrire sur le disque d'un client. Avec `Firefox`, les *cookies* sont dans

`C:\Documents and Settings\...\Application Data\Mozilla\Firefox\Profiles\xxxxxxx.default`

Avec `windows`, ils figurent dans

`C:\Documents and Settings\...\Cookies`

Le serveur peut ainsi gérer des sessions, mémoriser des préférences personnelles pour (et chez) le client ...

1.10.1 Envoyer un *cookie*

Avec la fonction `setcookie` :

```
int setcookie(string nom, string valeur, int date_expiration,
              string chemin, string domaine, int securite);
```

Tous les arguments sont optionnels, sauf le premier. Le domaine permet de limiter la visibilité à un site, le chemin d'accès restreint la visibilité à une partie de l'arborescence . Si la date d'expiration n'est pas précisée, le *cookie* disparaît à la fin de la session.

Lors d'une connexion à un site, le navigateur envoie automatiquement les cookies qui concernent le site (en fonction du domaine et du chemin d'accès). En `PHP` il est possible de connaître la liste des cookies et de leurs valeurs avec le tableau associatif `$_COOKIE[]`. Chaque cookie est accessible avec son nom comme nom de variable.

Exemple

```
<html>
<head><title> Cookies en PHP </title></head>
<body>
<h1>Cookies</h1>

<?php
setcookie("TestCookie","Valeur de test",time()+36000);
setcookie("T[0]","T1",time()+36000);
setcookie("T[1]","T2",time()+36000);
setcookie("T[2]","T3",time()+36000);

// Afficher un cookie
echo $_COOKIE["TestCookie"]."<br />";
echo $_COOKIE["T"]."<br />";

// Une autre méthode pour afficher tous les cookies
print_r($_COOKIE);
?>
```

```
</body>
</html>
```

Attention, la valeur du *cookie* ne s'affiche que si le *cookie* a déjà été déposé lors d'une session antérieure.

La valeur ne doit pas occuper plue de 4096 caractères.

FIGURE 1.10 – *Les cookies : affichage de la page au premier, puis au second chargement.*

1.11 Envoi d'un *mail*

C'est possible en `PHP` avec la fonction `mail`. Cette fonction est parfois désactivée chez certains hébergeurs.

```
<html>
<head>
<title>essai mail</title>
</head>
<body>
 <h4>Envoi de mail avec PHP</h4>
<?php
$dest = "claude.gueganno@free.fr"; /* destinataire */
$sujet = "essai de mail"; /* sujet */
$contenu = "Ce message a été envoyé automatiquement...\n";
$enTete = "From: claude.gueganno@ac-rennes.fr\n";
mail($dest, $mail, $sujet, $contenu, $enTete);
?>
</body>
</html>
```

Chapitre 2

Bibliothèques de fonctions PHP

Ce chapitre ne développe que les fonctions les plus communément utilisées.

2.1 Communication avec le client

2.1.1 Envoi de données vers le navigateur

Ces fonctions qui rapellent les fonctions classiques d'entrées/sorties standards on pour but d'envoyer le code HTML vers le navigateur.

`echo` : envoi de chaînes et de paramètres :

Exemple : `echo "<b>".$i."</b>";`
Le `.` permet de concaténer les différents éléments (constantes ou variables). Il peut être évité ; ainsi, le code

```
<?php
$nom = "Zaza";
$age = 99;
echo "nom : $nom ; age : $age";
?>
```

affiche `nom : Zaza ; age : 99`

`print` : envoi de chaînes :

Exemple : `echo "<h1>PHP</h1>";`

`printf` : envoi de chaînes formatées :

Exemple :

```
printf("Nom : %20s -- Age : %d<br>\n", $nom, $age);
```

Dans cet exemple , `%20s` et `%d` sont des formats associés aux variables respectives `$nom` (chaîne de caractères) et `$age` (entier).

Formats :

`d`	entier décimal
`c`	caractère ASCII
`o`	entier octal
`s`	chaîne
`x`	entier hexadécimal (lettres en minuscules)
`X`	entier hexadécimal (lettres en majuscules)
`f`	double
`b`	entier binaire
`e`	double en notation scientifique

Un format peut contenir des informations optionnelles :
- `-` pour justifier à gauche
- un nombre pour fixer le nombre de caractères à afficher
  Exemples : `%-5d` affiche un entier sur 5 caractères justifiés à gauche. `2.3f` pour un réel fixe à 2 le nombre de chiffres affichés avant la virgule et à 3 ceux qui sont après.

2.1.2 Envoi de données contenues dans un fichier

Voir le paragraphe 2.4.4 page 39.

2.2 Les chaînes de caractères

2.2.1 Séquences d'échappement

Elles permettent d'afficher les caractères spéciaux ou réservés :

`\n`	retour à la ligne
`\t`	tabulation
`\\`	caractère \
`\$`	caractère $
`\"`	caractère "

Exemples :
- `echo "<body bgcolor=\"red\">"` → `<body bgcolor="red">`
- `echo "prix : 4\$"` → `prix : 4 $`

`eval(chaîne)`	évalue la chaîne comme si c'etait du code `PHP`
`entier strlen(ch)`	retourne la longueur de *ch*
`tab count_chars(ch)`	retourne le nombre d'occurrences de chaque caractère de la chaîne (`PHP4`)
`chaîne strtok(ch, sep)`	separe *ch* en fonction de *sep*
`chaîne quotemeta(ch)`	retourne la chaîne avec un \ devant les caractères `. \ * ? [ ^ ] ( $ )`
`chaîne str_repeat (ch, n)`	répète *n* fois la chaîne (`PHP4`)

Exemple avec `strtok` , cette fonction permet de parcourir une chaîne en la découpant en sous-chaînes selon les occurences d'un séparateur. Dans l'exemple suivant, on choisit un découpage en mots. le séparateur est donc l'espace.

```
<html>
<head>
<title> Chaînes en PHP </title>
</head>

<body>
 <h1>Chaînes</h1>
<?php
$ch = "Ceci est une chaîne exemple";
echo "Chaîne : ".$ch."<br>";
echo "<b>strok</b><br>";
$sep = " ";
$mot = strtok($ch,$sep);
while($mot) { /* tant qu'il y a qq chose ... */
    echo "Mot = ".$mot."<br>";
    $mot = strtok($sep);
}
?>
</body>
</html>
```

Au premier appel, la fonction `strtok` est appelée avec 2 arguments. Ensuite, on utilise seulement le séparateur pour parcourir la chaîne. Le résultat du script est donné par la figure 2.1.

FIGURE 2.1 – *Fonction* ***`strtok`***.

2.2.2 Fonctions de conversion

`chaîne sprintf(format, arg)`	retourne une chaîne formatée (voir `printf`)
`chaîne chr(entier)`	donne le caractère qui correspond au code ASCII
`int ord(c)`	donne le code ASCII qui correspond au caractère c

2.2.3 Recherche de caractères et sous-chaînes

`entier strpos(ch1, ch2)`	retourne la position de la première occurrence de $ch2$ dans $ch1$
`entier strrpos(ch, car)`	retourne la position de la dernière occurrence du caractère
`entier strspn(ch1, ch2)`	longueur de la sous-chaîne de $ch1$ dont les caractères sont entièrement dans $ch2$
`chaîne strstr(ch1, ch2)`	retourne la portion de $ch1$ à partir de la première occurrence de $ch2$ et jusqu'à la fin
`chaîne stristr(ch1, ch2)`	`strstr` non sensible à la casse
`chaîne substr(ch, i, n)`	renvoie la sous-chaîne de taille n qui commence à l'indice i

2.2.4 Comparaison alphabétique

`entier strcmp(ch1, ch2)`	retourne 0 si $ch1 = ch2$, un nombre négatif si $ch1 < ch2$, un nombre positif si $ch1 > ch2$
`entier strcasecmp(ch1, ch2)`	comme `strcmp` sans tenir compte de la casse

2.2.5 Majuscules/minuscules

`chaîne strtolower(ch)`	convertit en minuscules
`chaîne strtoupper(ch)`	convertit en majuscules
`chaîne ucfirst(ch)`	met la première lettre en majuscule
`chaîne ucwords(ch)`	met tous les mots de ch en majuscules

2.2.6 Espaces

`chaîne chop(ch)`	retourne la chaîne sans les espaces
`chaîne trim(ch)`	supprime les espaces de début et fin de chaîne

2.2.7 Fonctions spéciales HTML

`chaîne htmlspecialchars (ch)`	convertit les caractères spéciaux en entités html < → < > → > & → & " → "
`chaîne htmlentities(ch)`	comme la fonction précédente, sauf que tous les caractères ayant une traduction HTML sont convertis
`chaîne nl2br(ch)`	ajoute devant chaque nouvelle ligne du texte
`parse_str(requete)`	analyse la requête comme si elle venait d'un formulaire posté par `get` (crée les variables et leur valeur)

2.2.8 Expressions rationnelles

Caractères spéciaux à utiliser

`|`	ou
`*`	0 ou plus
`+`	au moins une fois
`?`	0 ou une fois
`{`n`}`	n fois exactement
`{`n`,}`	n fois ou plus
`{`n`,`m`}`	au moins n et au plus m
`.`	un caractère quelconque
`^`	correspondance au début
`$`	correspondance en fin
`[a-z]`	tout caractère minuscule
`[ab]`	a ou b
`[^ab]`	tout sauf a et b
`[:alpha:]`	type de caractère (`alnum`, `blank`, `digit`, `punct`)
`[:<:]c`	mot commence par c
`[:>:]c`	mot finit par c

Les caractères spéciaux (`^ .[]()|?{}\`) utilisés comme données s'obtiennent en ajoutant un `\` devant.

Fonctions

`booléen ereg(exp,ch,tabocc)`	évalue l'expression et met les occurrences rencontrées dans *ch* dans le tableau *tabocc* la case 0 contient l'occurrence de l'expression, la case 1 la sous-chaîne. Retourne 0 s'il n'y a pas de correspondance
`chaîne ereg_replace(exp,rep,ch)`	remplace les sous-chaînes de *ch* qui correspondent à l'expression *exp* par la chaîne *rep*. *exp* peut être un code *ascii* **Exemple :** `ereg_replace(10, " ", ch)` remplace les `\n` par des ` `
`booléen eregi(exp,ch,tab)`	comme `ereg` sans tenir compte de la casse
`booléen eregi_replace(exp,rep,ch)`	comme `ereg_replace` sans tenir compte de la casse
`tableau split(exp,ch,limite)`	retourne un tableau des sous-chaînes

```
<html>
<head>
<title> ereg </title>
</head>

<body>
 <h3>Expressions rationnelles</h3>

<?php
 echo "Chaîne originale : ".$HTTP_USER_AGENT."<br>\n";
 ereg("^([A-Za-z]+)/.*$", $HTTP_USER_AGENT, $T);
 echo "navigateur = ".$T[1]."<br>";
?>
</body>
</html>
```

FIGURE 2.2 – *Fonction* `ereg` *exécution avec Konqueror.*

FIGURE 2.3 – *Fonction* `ereg` *exécution avec Netscape.*

2.3 Fonctions mathématiques

Trigonométrie

`décimal acos(x)`	arc cosinus
`décimal asin(x)`	arc sinus
`décimal atan(x)`	arc tangente
`décimal cos(x)`	cosinus
`décimal sin(x)`	sinus
`décimal tan(x)`	tangente de l'angle x

Autres fonctions

`nombre abs(x)`	valeur absolue ($\|x\|$)
`décimal exp(x)`	exponentielle (e^x)
`décimal log(x)`	logarithme
`décimal pi()`	retourne la valeur de π
`décimal pow(x, y)`	élève x à la puissance y (x^y)
`décimal sqrt(x)`	racine carrée de x ($\sqrt{x}$)

Conversions

`chaîne decbin(entier)`	retourne la représentation binaire de `entier`
`entier bindec(chaîne)`	retourne la représentation décimale de la chaîne binaire
`chaîne dechex(entier)`	retourne la représentation hexadécimale de `entier`
`chaîne decoct(entier)`	retourne la représentation octale de *entier*
`décimal deg2rad(angle)`	conversion degré → radian
`entier hexdec(chaînehex)`	conversion chaîne hexadécimale en un entier
`décimal octdec(chaîneoct)`	conversion chaîne octale en un entier
`décimal rad2deg(angle)`	conversion radian → degré

Arrondis

`nombre abs(x)`	valeur absolue
`entier ceil(x)`	retourne le plus petit entier $> x$
`entier floor(x)`	partie entière
`entier round(val)`	arrondi à l'entier le plus proche

nombres aléatoires

`entier getrandmax()`	renvoie le nombre aléatoire maximum pour `rand`
`entier rand(inf, sup)`	renvoie un nombre aléatoire compris entre les entiers inf et sup
`entier srand(entier)`	pour initialiser le générateur de nombres aléatoires

2.4 Les fichiers

2.4.1 Ouverture, fermeture

Ouverture

```
$f = fopen("nom","mode");
```

La fonction renvoie un pointeur sur le fichier. En cas d'échec, la valeur 0 est retournée. Cette fonction permet aussi d'établir une connexion `HTTP` ou `FTP`.

Exemples :

```
$fp = fopen("/home/rasmus/file.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@example.com/", "w");
```

Sous `Windows`, le séparateur \ doit être doublé (caractère spécial).

```
$fp = fopen("c:\\data\\info.txt", "r");
```

Modes d'ouvertures :

`"r"`	lecture
`"w"`	écriture
`"a"`	ajout
`"r+"`	lecture et écriture
`"w+"`	lecture et écriture, supprime le contenu précédent
`"a+"`	lecture et écriture en fin de document

Remarque : pour un fichier binaire il faut ajouter après le mode un `b`, par exemple, `"rb"` ...

Traitement des ereurs

```
if($fp == 0){
  echo "erreur d'ouverture du fichier";
  exit; /* fin du script */
}
```

Ouverture et stockage dans un tableau : ces 2 opérations sont enchainées par la fonction `file`. Chaque élément du tableau est une ligne du fichier terminée par `'\n'`.

```
$ft = file("nom.txt");
for($i = 0; $i < count($ft); $i++) {
   print($ft[$i]);
}
```

Fermeture

```
fclose($fp);
```

2.4.2 Lecture et écriture

`booléen feof(fp)`	Renvoie 0 lorsque la fin de fichier est atteinte
`chaîne fgets(fp, n)`	Lecture de $n-1$ caractères dans le fichier fp. La lecture s'arrête à la fin de fichier ou à la fin de ligne ou à $n-1$ caractères.
`car fgetc(f)`	Lecture d'un caractère
`fputs(fp,str)`	Écriture dans le fichier fp de la chaîne str.
`fputc(fp,c)`	Écriture du caractère c

2.4.3 Manipulation de fichiers et répertoires

Modifications (nom et droits d'accès)

`booléen chgrp("nomfich","nomgroupe")`	change le groupe d'appartenance
`booléen chmod("nomfich",mode)`	change les droits d'accès
`booléen chown("nomfich","propriétaire")`	change le propriétaire.
`int rename("nomfich1", "nomfich2")`	change le nom du fichier.

Informations sur les attributs des fichiers

`chaîne basename(chemin)`	retourne le nom de fichier d'un chemin
`entier fileatime("nomfich")`	temps écoulé depuis le dernier accès au fichier.
`entier filesize("nomfich")`	taille en octets du fichier.
`booléen file_exists("nomfich")`	1 si le fichier existe.
`booléen is_file("nomfich")`	1 si nomfich est un fichier.
`booléen is_executable("nomfich")`	1 si fichier existe et est en mode exécutable.
`booléen is_readable("nomfich")`	1 si fichier existe et est en mode lecture.
`booléen is_writeable("nomfich")`	1 si fichier existe et est en mode écriture.
`chaîne tempnam(rep, ch)`	création d'un fichier temporaire unique dans le répertoire rep. Le préfixe est optionnel, retourne le nom du fichier temporaire.

Gestion des répertoires

`chaîne dirname("chemin")`	retourne le répertoire d'un chemin
`booléen is_dir("nom")`	1 si nom est un répertoire.
`booléen chdir("repertoire")`	change de répertoire, renvoie 1 si ok.
`closedir(pointeur)`	ferme un répertoire ouvert par `opendir`
`booléen copy("source","destination")`	copie d'un fichier vers un répertoire.
`booléen mkdir("nomrep")`	crée un nouveau répertoire
`entier opendir("nomrep")`	ouvre un répertoire.
`booléen rmdir("nomrep")`	supprime le répertoire.
`chaîne readdir(pointeur)`	pour lire un élément du répertoire (retourne un nom de fichier ou de répertoire). Le pointeur est retourné par `opendir`.
`rewinddir(pointeur)`	revenir au début du répertoire.

Exemple d'affichage de répertoires .

```
<html>
<head>
<title> readdir </title>
</head>

<body>
<h3>Répertoire /</h3>
<?php
$rp = opendir("/");
while($x = readdir($rp)) echo $x." ";
closedir($rp);
?>
</body>
</html>
```

L'exécution du script est donné par la figure 2.4. C'est bien la racine du système qui est prise en compte et non la racine du site *web*. Dans le cas d'un service *internet*, il est donc important de soigner la sécurité des fichiers et des répertoires.

FIGURE 2.4 – *Fonction* ***readdir****.*

2.4.4 Envoi de fichiers vers le navigateur

`fpassthru` : le contenu du fichier est envoyé vers le navigateur en une seule instruction.

```
$fp = fopen("fich.html", "r");
if ($fp) {
  fpassthru($fp);
```

```
  fclose($fp);
}
```

`readfile` : Lit le fichier et l'envoie au navigateur, retourne le nombre d'octets lus.

```
readfile("fich.html");
```

L'ouverture du fichier, sa lecture et l'envoi vers la sortie standard (le navigateur) sont enchaînés. Comme pour `fopen`, le nom de fichier peut commencer par `http://` ou `ftp://`.

2.5 Réseau

2.5.1 Fonction `fsockopen`

Cette fonction permet d'établir une connexion avec un serveur sur un port donné.

```
$fp  = fsockopen($serveur, $port ,[&$errno, &$errstr, $timeout]);
```

Les trois derniers paramètres sont optionnels.

Exemple : connexion sur un serveur `POP`. Le numéro de port attribué au service `POP` est le 110. Le protocole est relativement simple (identification avec les requètes `USER` et `PASS`, puis interrogation du serveur).

Dans l'exemple, on réalise une connexion avec le serveur, puis on lui demande le nombre de messages en attente (requète `STAT`).

Les paramètres et les fonctions utiles ont été encapsulées dans une classe `message` :

```
/* Fichier 'message.php' */
<?php
class message {
    var $server; // nom du serveur
    var $user;   // utilisateur
    var $pass;   // mot de passe
    var $stream; // pointeur pour la communication

    // Constructeur
    function message($s,$u,$p){
        $this->server=$s;
        $this->user=$u;
        $this->pass=$p;
    }

    /* ouverture du port */
    function open(){
        $this->stream  = fsockopen($this->server, 110,&$errno, &$errstr, 30);
        if(!$this->stream) return 0; else return 1;
    }

    /* Connexion => lecture de l'en-tête + user + password */
    function connect(){
        /* Lecture en-tête */
        $c = fgetc($this->stream);
        $b = $c.fgets($this->stream, 128);
        echo "<br /> b = ".$b."<br />";
        if (!strpos($b, "OK")) return false;
        /* Nom d'utilisateur */
        fputs($this->stream,"USER ".$this->user."\n");
        $b = fgets($this->stream, 128);
        echo $b."<br>";
        if (!strpos($b, "OK")) return false;
        /* mot de passe */
```

```
        fputs($this->stream,"PASS ".$this->pass."\n");
        $b = fgets($this->stream, 128);
        if (strpos($b, "OK") != 1) return false;
        echo $b."<br>";
        return true;
    }

    /* Nombre de messages sur le serveur */
    function stat() {
        fputs($this->stream,"STAT\n");
        $b = fgets($this->stream, 128);
        if (strpos($b, "OK") != 1) return false;
        return $b;
    }

    function close(){
        fputs($this->stream,"QUIT");
        fclose($this->stream);
    }
}
?>
```

L'utilisation de la classe est donnée dans le script principal :

```
<html>
<body>
<h3>fsockopen</h3>
<?php
require('message.php'); // classe pour 'pop'
echo "<h2>Local</h2>";
$user = "   ";   // à remplir
$motdepasse = "    "; // à remplir
$M = new message("pop.free.fr",$user,$motdepasse);
echo "open:".$M->open()."<br />";
echo "connect:".$M->connect()."<br />";
echo "nombre de message(s) = ".$M->stat();
$M->close();
?>
</body>
</html>
```

2.5.2 Fonction `header`

Cette fonction sert à modifier l'en-tête `HTTP` du fichier `HTML` envoyé vers le navigateur. Les appels doivent être faits avant les sorties `html`.

Redirection d'adresse. La modification du champ `Location` de l'en-tête permet de rediriger le navigateur vers un autre site.

```
header("Location: http://www.autresite.fr");  /* Redirection du navigateur */
exit;  /* Rien ici après la redirection */
```

Désactivation des caches. Lorsqu'un script PHP génère du code html qui ne doit pas être pris en compte par le cache d'un navigateur ou d'un *proxy*, il faut également intervenir sur l'en-tête HTTP du fichier :

```
/* Expires -> on donne une date du passé ... */
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
/* Last-Modified -> modifié à l'instant
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-cache, must-revalidate");  // pour HTTP/1.1
header("Pragma: no-cache");                          // pour HTTP/1.0
```

Avec un tel en-tête, le fichier sera automatiquement rechargé par le navigateur et/ou les serveurs *proxies* intermédiaires entre le serveur et le navigateur.

Utilisation pour le WAP. L'utilisation des scripts PHP pose un problème de nom de fichier dans le cas du WAP : les navigateurs WAP ne veulent pas des fichiers dont l'extension est autre que *.wml*. La solution consiste alors à modifier l'en-tête du fichier avant l'envoi vers le navigateur, comme dans l'extrait suivant :

```
/* fichier 'test.php' */
<?php
   $f = fopen("foo.wml","w");
   fputs($f,"<?xml version=\"1.0\"?>\n");
   fputs($f,"<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"".
         "\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");
   fputs($f, "<wml>\n<card id=\"rep\">\n");
     ...
     ...
   fputs("</card>\n</wml>\n");
   fclose($f);
   header("Location: http://claude.gueganno.free.fr/wap/foo.wml")
   exit;
?>
```

Ceci permet d'utiliser la technologie PHP vers un terminal mobile.

2.6 Date et gestion du temps

`chaine date(format)` : retourne la date

a	am ou pm	d	jour du mois sans les zéros
D	jour de la semaine en 3 lettres	F	nom du mois
h	heure de 1 à 12	H	heure de 0 à 23
m	minutes	j	jour du mois avec les zéros
l	jour de la semaine	m	chiffre du mois
M	nom du mois en abrégé	y	année sur deux chiffres
Y	année sur 4 chiffres	z	jour de l'année
t	nombre de jours du mois	z	jour de l'année

Exemple : `echo date("a D h m l M Y d F H j m y z")` affiche :

`pm Tue 03 08 Tuesday Aug 2003 19 August 15 19 08 03 230`

`tableau gettimeofday();` : renvoie la date courante sous la forme d'un tableau associatif. Les clés principales sont :

- `"sec"` : donne le nombre de secondes écoulées depuis le 1er janvier 1970. Cet entier long est une représentation de la date appelée «*timestamp*».
- `"usec"` : donne le nombre de microsecondes écoulées dans la seconde courante.

`tableau getdate(timestamp);` : renvoie un tableau associatif correspondant à la date donnée en argument. Les clés sont données dans le tableau suivant :

`"seconds"`	secondes	`"minutes"`	minutes
`"hours"`	heures	`"mday"`	jour du mois
`"wday"`	jour de la semaine (numérique)	`"mon"`	mois (numérique)
`"year"`	année (numérique)	`"yday"`	jour dans l'année
`"weekday"`	jour dans la semaine (textuel)	`"month"`	mois textuel

`entier mktime(h,m,s,mois,j,a)` : renvoie un entier long : le *timestamp*.

Exemple (`date`, `getdate`, `mktime`)

```
<html>
<head>
<title> date </title>
</head>

<body>
<?php
echo "<h4>Construction d'une date formatée (<code>date()</code>)</h4>";
echo "Heure locale :".date("M d Y H:i:s", mktime(1,2,3,4,5,2003));
echo "<br>";
echo "Heure GMT :".gmdate("M d Y H:i:s", mktime(1,2,3,4,5,2003));
echo "<h4>Heure (<code>gettimeofday()</code>)</h4>";
$T = gettimeofday();
echo "Secondes : ".$T["sec"]."  microsecondes : ".$T["usec"];
echo "<h4>Date (<code>getdate()</code>)</h4>";
$T = getdate();
echo "secondes : <b>".$T["seconds"]."</b><br>";
echo "minutes: <b>".$T["minutes"]."</b><br>";
echo "heures : <b>".$T["hours"]."</b><br>";
echo "jour du mois : <b>".$T["mday"]."</b><br>";
echo "jour de la semaine : <b>".$T["wday"]."</b><br>";
echo "mois : <b>".$T["mon"]."</b><br>";
echo "année : <b>".$T["year"]."</b><br>";
echo "jour de l'année : <b>".$T["yday"]."</b><br>";
echo "jour de la semaine : <b>".$T["weekday"]."</b><br>";
echo "mois : <b>".$T["month"]."</b><br>";
?>
</body>
</html>
```

L'affichage produit est donné par la figure 2.5.

FIGURE 2.5 – *Date et heure.*

`setlocale("LC_TIME", "fr_CA");` : avec ces paramètres, on obtient la possibilité d'afficher les paramètres textuels en français, avec la fonction `strftime`.

`chaîne strftime(format, [timestamp]);` : renvoie la date sous la forme d'une chaîne formatée. Si le paramètre *timestamp* est ignoré, c'est la date du jour qui est prise en compte. Les formats sont donnés dans le tableau suivant :

`%a`	jour de la semaine abrégé
`%A`	jour de la semaine
`%b`	mois en abrégé
`%B`	mois
`%c`	représentation complète date et heure
`%d`	jour du mois (de 00 à 31)
`%H`	heure (de 00 à 23)
`%I`	heure (de 01 à 12)
`%j`	jour dans l'année (de 001 à 366)
`%m`	mois (de 1 à 12)
`%M`	minute
`%p`	`'am'` ou `'pm'`
`%S`	second as a decimal number
`%U`	numéro de semaine (commençant le dimanche)
`%W`	numéro de semaine (commençant le lundi)
`%w`	jour de la semaine (dimanche = 0)
`%x`	date
`%X`	heure
`%y`	l'année sans le siècle (de 00 à 99)
`%Y`	l'année et le siècle (ex. : 2003)
`%Z`	zone horaire
`%%`	le caractère `'%'`

`entier time()` : retourne la valeur du *timestamp* courant.

Exemple (`setlocale`, `strftime`)

```
<html>
<head>
<title> date </title>
</head>

<body>
<?php
```

```
setlocale ("LC_TIME", "fr_CA");
print(strftime("%A %d %B %Y", mktime(1,1,1,7,14,2003)));

echo "<br> Nous sommes le ".strftime("%A %d %B %Y")."<br>";
echo "Dans 360 jours nous serons le ";
$td = mktime(0,0,0,date("m"),date("d")+360,date("Y"));
echo strftime("%A %d %B %Y", $td);
echo "<td>";
?>
</body>
</html>
```

L'affichage produit est donné par la figure 2.6. On remarque la robustesse de la fonction `mktime` qui accepte ici un numéro de jour > 7.

FIGURE 2.6 – *Date et heure en français.*

Chapitre 3

Les sessions

3.1 Présentation

Les sessions en PHP permettent de sauvegarder des variables de page en page pendant une certaine durée gérée dans les programmes PHP.

L'avantage des sessions sur les variables de type $_GET et $_POST est double : outre l'absence de formulaires (et même de tout code HTML) pour gérer les sessions, ce système permet de transmettre des variables sur toutes les pages du site d'une manière transparente.

Les variables de session vont pouvoir mémoriser toutes les données que l'on souhaite lors connexion d'un visiteur.

Applications : les sessions peuvent servir à quel moment un visiteur (ou une adrese IP) est connecté(e) sur un site, et pendant combien de temps (utile pour les discussions en ligne). On utilise également les sessions pour gérer les paniers dans les sites marchands.

3.2 Principe

Chaque utilisateur ayant besoin des sessions se voit attribuer un identifiant unique appelé *id* de session. Cet identifiant est stocké sur le poste de l'internaute sous forme d'un *cookie* ou transite via l'URL si l'option session.use_trans_sid est à 1 (ou «On») dans le fichier *php.ini*. La méthode du *cookie* est la plus recommandée.

Avant d'utiliser les sessions sur une page, on doit toujours utiliser la fonction session_start() placée avant tout envoi de code HTML, et donc généralement tout en haut de la page PHP :

```
<?php
     session_start();
?>
```

Importance de la place de cette instruction. PHP utilise les *cookies* pour repérer quel est l'*id* de session utilisé par l'internaute. Or, le protocole

HTTP fonctionne de telle sorte que les en-têtes (qui permettent de dire au navigateur de créer un *cookie*) sont envoyés avant le premier caractère HTML transmis. Cela veut dire que dès que vous transmettrez un caractère HTML, les en-têtes seront envoyés et vous ne pourrez plus les modifier, vous ne pourrez donc plus écrire le *cookie* de session.

3.2.1 Variables de session

Elles sont contenues dans le tableau global `$_SESSION[]`. Les variables de session fonctionnent comme les variables classiques. Voici un exemple pour attribuer une valeur à une variable de session nommée `login` :

```
<?php
     session_start(); // toujours ..
     $_SESSION['login']='valeur';
?>
```

Pour récupérer la valeur d'une variable de session (sur la même page ou sur une autre page, après un `session_start()`), il s'agit de la même procédure que pour une variable classique hormis l'ajout du `session_start()` tout en haut de la page :

```
<?php
     session_start();
     if(isset($_SESSION['login'])) {
          echo $_SESSION['login'];
     }
?>
```

Pour savoir si une variable de session existe, on procède de la même façon que pour les autres variables à savoir qu'on utilise la fonction `isset()`.

Pour effacer une ou plusieurs variables de session, au même titre qu'une autre variable en PHP, il faut utiliser pour la fonction `unset()` :

```
<?php
     session_start();
     if(isset($_SESSION['login']))
     {
          unset($_SESSION['login']);
     }
?>
```

3.2.2 Fin de session

Pour effacer toute la session d'un coup, il faut utiliser la fonction `session_destroy()` qui détruit toutes les variables de session d'un visiteur.

Pour une plus grande sécurité, on peut avoir envie de connaître l'*id* de session du visiteur, pour pouvoir effectuer divers traitements dessus ou encore pour l'enregistrer quelque part pour en garder une trace. PHP dispose d'une fonction destinée à cet usage, il s'agit de la fonction `session_id()`.

```
<?php
     session_start();
```

```
     echo session_id(); //Retourne l'identifiant de session
?>
```

Il est possible de spécifier l'*id* de session courant en lui affectant une valeur grâce à la fonction `session_id()`. Ceci doit être fait avant le `session_start()`.

```
<?php
     session_id(md5(mt_rand()));
     session_start();
     echo session_id(); //Retourne l'identifiant de session
?>
```

3.3 Exemple

Une première page propose un choix de produits. Sélectionner un produit, c'est le mettre dans le «panier». Le panier est une chaîne de caractères stockée dans une variable de session. La figure 3.1 donne un scénario possible.

3.3.1 Page principale

- Le premier appel est `session_start()`.
- Si la variable `$_SESSION["panier"]` n'existe pas elle est initialisée, à partir d'une donnée prise dans le tableau `$_GET[]`.
- La variable `$_GET[]` est détruite avec `unset()`, pour éviter qu'au rechargement de la page, le produit soit à nouveau ajouté au panier.
- L'appel à `header()` permet d'éliminer le paramètre `?produit=xx` de l'URL.

```
<?php
     session_start();
     if (isset($_GET["produit"])) {
          if (!isset($_SESSION["panier"])) $_SESSION["panier"] = $_GET["produit"];
          else $_SESSION["panier"] .= ",".$_GET["produit"];
          unset($_GET["produit"]);
          header("Location: http://localhost/php/session/session1.php");
     }
?>
<html lang="fr">
      <head>
          <title>les sessions</title>
      </head>
      <body>
          <ul>
               <li><a href="session1.php?produit=1">Acheter le produit 1</a></li>
               <li><a href="session1.php?produit=2">Acheter le produit 2</a></li>
               <li><a href="session1.php?produit=3">Acheter le produit 3</a></li>
               <li><a href="session1.php?produit=4">Acheter le produit 4</a></li>
               <?php
                    if (isset($_SESSION["panier"]) ) {
                         echo "<li><a href=\"session2.php\">Commander</a></li>";
                    }
               ?>
```

```
            </ul>
            <?php
              if (isset($_SESSION["panier"]) ) {
                 echo "<h1>Panier</h1>";
                 echo $_SESSION["panier"];
              }
            ?>
        </body>
</html>
```

3.3.2 Page finale

L'appel à cette deuxième page montre bien la pérennité des variables de session. Pour recommencer à zéro, il faut appeler `session_destroy()`

```
<?php session_start(); ?>

<html>
      <head>
          <title>les sessions</title>
      </head>
      <body>
          <p class="">Vous avez command&eacute; : <br />
              <?php if (isset($_SESSION["panier"])) echo $_SESSION["panier"];
                    echo "<br />Fin de la session ".session_id()."<br />";
                    session_destroy();
              ?>
            <br />
            <a href="session1.php">D&eacute;marrer une nouvelle session</a>
          </p>
      </body>
</html>
```

FIGURE 3.1 – *Exemple avec les sessions*

Chapitre 4

Base de données

PHP propose des fonctions pour accéder à plusieurs bases de données (mysql, mSQL, PostgreSQL, Informix, Oracle, access, ...). Selon la base choisie, l'interface est plus ou moins étoffée, mais on retrouve à chaque fois des fonctions pour :

- se connecter au serveur de base de donnée ;
- se lier à une base du serveur ;
- effectuer une requète ;
- exploiter le résultat de la requète dans un tableau ou séquentiellement ;
- se déconnecter.

Le paragraphe suivant donne un aperçu de l'utilisation d'une base de données avec mysql, base largement utilisée sur le *web*.

4.1 Connexion au serveur et sélection de la base

entier `mysql_connect(hostname, user, password)` : ouvre une connexion avec le serveur. Un entier > 0 est renvoyé en cas de réussite. En cas d'échec, la fonction retourne 0. Si deux appels sont faits à la fonction, il n'y a pas 2 connexions simultanées : c'est toujours la même qui court. La connexion est annulée à la fin du script, ou après un appel à `mysql_close`.

`entier mysql_select_db(nom_base, [lien]);` : sélectionne une base de donnée sur le dernier lien ouvert avec `mysql_connect`. Le paramètre *lien* est donc facultatif.

Renvoie *vrai* en cas de réussite et $faux$ sinon.

Exemple :

```
function connecter(){
     $host="localhost";
     $user="netcr";
     $password="stsii2";
     $database="netcr";

     /*connection a la base de données*/
```

```
        $lien = mysql_connect($host, $user, $password)
                or die("erreur de connexion");
        mysql_select_db($database, $lien)
                or die("erreur de table");
}
```

Dans la plupart des cas, le choix de l'hôte `localhost` est correct, puisque le script `PHP` s'exécute sur le serveur.

`entier mysql_list_tables(nomBase,[lien]);` : renvoie un pointeur sur la liste des noms de tables de la base identifiée par la chaîne *nomBase*.

`chaîne mysql_tablename(resultat, i);` : donne le nom de la $i^{\text{ème}}$ table de la base. *resultat* est un paramètre obtenu par un appel à `mysql_list_tables()`.

Exemple :

```
<?php
mysql_connect ("localhost:3306");
$resultat = mysql_list_tables ("netcr");
$i = 0;
while ($i < mysql_num_rows($resultat)) {
  $T[$i] = mysql_tablename ($resultat, $i);
  echo $T[$i] . "<BR>";
  $i++;
}
?>
```

4.1.1 Requète

`entier mysql_query(requete, [lien]);` : effectue une requète `SQL` sur la base sélectionnée du dernier lien ouvert. Le paramètre *lien* est facultatif. La requète est une chaîne de caractères qui ne doit pas se terminer par le caractère `';'`.

Renvoie un numéro identificateur de résultat > 0 en cas de réussite , et $faux$ (0) sinon.

Exemple :

```
connecter();
$req = "select * from maTable where x<5";
$resultat = mysql_query($req)
or die ("Erreur de requète SQL : ".$req."<br>";
```

4.2 Exploitation du résultat de la requète

Pour ces fonctions, le paramètre en entrée est celui qui est retourné par `mysql_query`.

`tab mysql_fetch_array(res)`	Retourne un tableau associatif qui représente tous les champs d'une rangée du résultat. Chaque appel produit la rangée suivante jusqu'à la fin.
`tab mysql_fetch_row(res)`	Retourne un tableau qui représente tous les champs d'une rangée du résultat. Chaque appel produit la rangée suivante jusqu'à la fin.
`objet mysql_fetch_object(res)`	Renvoie un objet dont les propriétés sont les noms des champs. Chaque appel produit un objet corrspondant à la rangée suivante jusqu'à la fin.
`entier mysql_num_rows(res)`	Donne le nombre de lignes
`entier mysql_num_fields(res)`	
`entier mysql_affected_rows([lien])`	renvoie le nombre de rangées affectées par la dernière requète `INSERT`, `UPDATE` ou `DELETE`

4.3 Déconnexion

Elle est automatique à la fin de l'exécution du script. Elle peut être provoquée par un appel à `mysql_close()`.

4.3.1 Exemple

On dispose d'une base de données `'claude'` contenant une table table `'mesures'` composée de 4 champs. Il y a deux enregistrements dans cette table. Il s'agit, de se connecter à la base de données en PHP, puis de créer une page HTML qui affiche le contenu de la table `'mesures'`.

FIGURE 4.1 – *Base de données pour l'exemple*

```
$link = mysql_connect('localhost', 'claude', 'mdp');
if (!$link) {
   die('Impossible de se connecter : ' . mysql_error());
}

// sélection de la base 'claude'
$db = mysql_select_db('claude', $link);
if (!$db) {
   die ('Impossible de sélectionner la base de données : '.
        mysql_error());
}

$requete = "select * from mesures";
$resultat = mysql_query($requete);

if (!$resultat) {
```

```
   echo "Impossible d'exécuter la requète ($requete) dans la base : " .
         mysql_error();
   exit;
} else if (mysql_num_rows($resultat) == 0) {
   echo "Aucune ligne trouvée, rien à afficher.";
   exit;
} else {
  echo "<ul>";
  while ($row = mysql_fetch_assoc($resultat)) { // lecture séquentielle
     echo "<li>".$row["id"] ." : ".$row["label"]." = ".
          $row["valeur"]." (".$row["temps"].")";
  }
  echo "</ul>";
}

mysql_free_result($resultat);
```

FIGURE 4.2 – *Exécution du script.*

Chapitre 5

Autres bibliothèques

En dehors des fonctions présentées ici, notons également des bibliothèques de fonctions pour :

- travailler avec les `URL`,
- concevoir une image,
- concevoir un document `PDF`,
- crypter et décrypter un document,
- compresser et décompresser un document,
- accéder à un annuaire `LDAP`,
- envoyer un $e-mail$,
- consulter une boîte `POP` ou `IMAP`,
- ...

Une documentation complète peut être obtenue sur

`http://www.php.net`
`http://www.php.net/download-docs.php`

Index