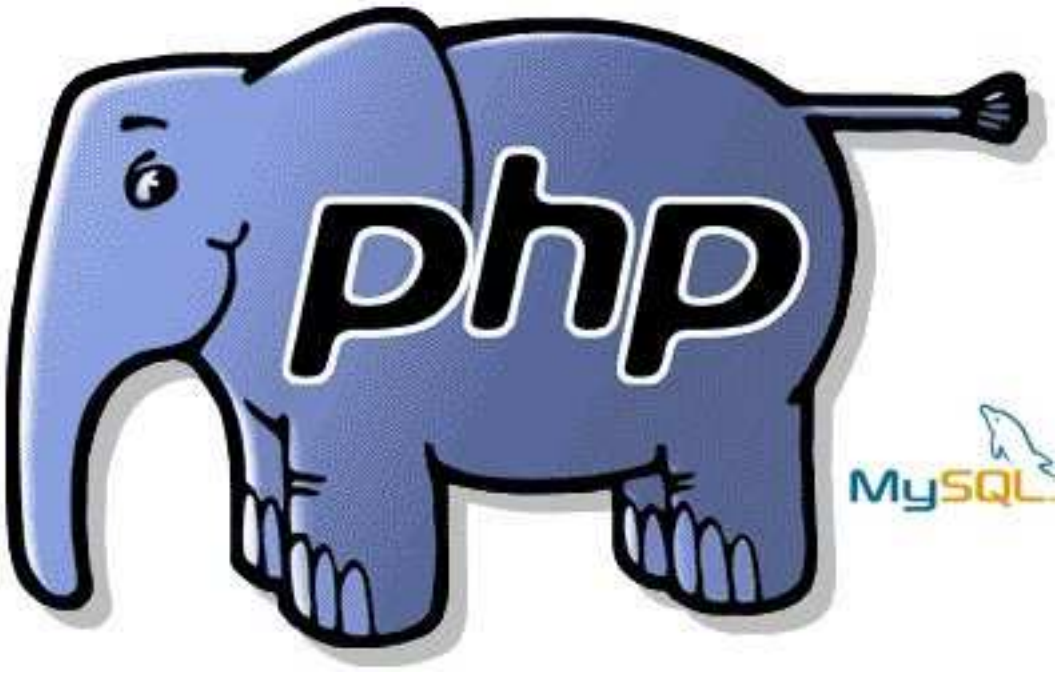
Tristan Colombo

Outils de l'Internet (côté Serveur)



Université d'Aix Marseille I - Provence
Laboratoire d'Informatique Fondamentale de Marseille

Couverture : logos PHP et MySQL

Outils de l'Internet (côté Serveur) – Version 1.2 : juin 2005

Tristan Colombo

Outils de l'Internet (côté Serveur)

Version 1.2

Le monde juge bien des choses,
car il est dans l'ignorance naturelle
qui est le vrai siège de l'homme.
Les sciences ont deux extrémités qui se touchent,
la première est la pure ignorance naturelle
où se trouvent tous les hommes en naissant,
l'autre extrémité est celle où arrivent les grandes âmes qui,
ayant parcouru tout ce que les hommes peuvent savoir,
trouvent qu'ils ne savent rien et se rencontrent en cette même ignorance
d'où ils étaient partis, mais c'est une ignorance savante qui se connaît.
Pascal

Table des matières

Introduction

De nos jours, avec le développement et la généralisation d'Internet, de plus en plus d'applications permettent de dialoguer avec des serveurs. Ces applications dynamiques exécutent des programmes sur un serveur en fonction de données fournies par le client. Ce dialogue s'effectue selon des règles très précises : le *protocole*. Le protocole du Web est HTTP, mais il peut être possible de communiquer avec un site via d'autres protocoles, comme par exemple FTP qui permet d'échanger des fichiers.

Avant que d'aborder plus précisément le mécanisme client/serveur et les différents langages, intéressons nous aux outils de développement. Ce sont ceux que vous allez utiliser pour développer vos applications et ils peuvent vous faire gagner un temps précieux ... ou vous le faire perdre ! Premier rappel élémentaire : vous allez produire du code HTML (que ce soit de manière directe ou indirecte) ; pensez donc à l'indenter correctement. Ceci pour deux raisons :

1. Pour vous : il est beaucoup plus simple de relire et débugger un code présenté de manière claire.
2. Pour l'utilisateur : si une de vos pages présente une particularité, un utilisateur peut avoir la curiosité d'afficher votre code ... ce qui peut produire le plus mauvais effet !

Pensez également qu'il existe une grande diversité de navigateurs web et que vos pages devront être affichées correctement sur tous ces types de navigateurs et ceci quelle que soit la résolution graphique de l'utilisateur. Ceci demande bien sûr de nombreux tests mais surtout de bannir l'utilisation de code non normalisé. Si vous utilisez des objets spécifiques tels que les cadres – *frames* – ou des applications en Flash, vous devez tester le navigateur client pour savoir s'il contient le plugin nécessaire et sera en mesure d'afficher correctement la page. Voici maintenant un aperçu non exhaustif des outils de développement disponibles qui seront notés à l'aide d'étoiles (de une pour "Très Mauvais" à cinq pour "Indispensable").

Environnements de développement intégrés

Il s'agit d'environnements contenant un éditeur avec coloration syntaxique. Ils peuvent contenir également des outils WISIWIG permettant de coller des images, insérer des animations, etc. Le code généré est plus ou moins "pur".

- **DreamWeaver – Windows – ★**
  Le plus connu des environnements de développement intégré orienté Web. Il est très convivial et graphiquement très joli. Il a pourtant de très grosses lacunes. Tout d'abord il n'est disponible que sous Windows (ce qui suffit à en faire un outil inutilisable ...). De plus le code généré est très mal indenté et contient un nombre impressionnant de ligne inutiles, ce qui le rend très rapidement illisible.

- **Quanta Plus – Linux – ★ ★**
  Equivalent de DreamWeaver sous Linux. Il produit un code un peu plus propre et mieux indenté. Les fonctionnalités disponibles sont les mêmes.
- **BlueFish Editor – Linux – ★ ★ ★ ★**
  Le plus complet des environnements : idéal pour ceux qui aiment ce genre d'outil. L'interface est assez austère mais bien conçue. Contient beaucoup plus de fonctionnalités que les outils de la gamme DreamWeaver. L'absence d'outils WISIWIG permet la production d'un code pur.
- **KDevelop – Linux – ★ ★**
  Très bon outil de développement généraliste ... mais justement trop généraliste pour être utilisé en développement internet.
- **Eclipse – Linux & Windows – ?**
  Environnement de développement complet. Défaut : développé notamment par Borland et IBM, risque de devenir payant à court terme.

Editeurs

Quelques éditeurs contenant une coloration syntaxique pour ceux préférant tout maîtriser et coder en "dur" (ce qui n'est pas une mauvaise idée ...).

- **NEdit – Linux – ★ ★**
  Editeur assez basique.
- **XEmacs – Linux – ★ ★ ★ ★**
  Editeur prenant en compte les expressions relationnelles. S'il vous faut absolument un éditeur en mode graphique, celui-ci représente à mes yeux le meilleur choix ... en mode graphique ...
- **VIm – Linux & Windows – ★ ★ ★ ★ ★**
  L'éditeur de référence en mode console!!! Il faut bien sûr passer un peu de temps pour apprendre les commandes de bases ... mais que de temps gagné par la suite!

Navigateurs

Voici quelques navigateurs qui pourront être utilisés pour tester l'affichage de vos pages. L'un d'entre eux pourra même être utilisé en phase de développement.

- **Internet Explorer – Windows – ★**
  Le plus connu des navigateurs car distribué obligatoirement avec toutes les versions de Windows. Mais comme beaucoup de produits Microsoft, il s'agit aussi du plus mauvais ...
- **Netscape Navigator – Linux & Windows – ★**
  La tentative de réponse de Netscape à Microsoft. Au final un produit équivalent.
- **Mozilla-FireFox – Linux & Windows – ★ ★ ★ ★ ★**
  Très bon navigateur permettant de bloquer les popups et auquel on peut intégrer de nombreuses extensions. Associé aux extensions "Web Developper" et "HTML Validator", il s'agit du meilleur navigateur orienté développement. Il permet de tester et de valider vos pages, de débugger le JavaScript, etc.

Systèmes de Gestion de Bases de Données

Voici deux systèmes libres pouvant être associés à PHP.

- **MySQL – Linux & Windows – ★ ★ ★ ★ ★**
  SGBD relationnel libre très répandu, respectant pratiquement intégralement la norme SQL92.
- **SQLite – Linux & Windows – ?**
  Nouveauté de la version 5 de PHP : intégration par défaut du support de SQLite. Ce système serait particulièrement adapté aux petites bases de données (sans remettre en cause l'utilisation de MySQL). Contrairement à un vrai SGBD, ce système ne gère pas les noms d'hôte, login, et mots de passes. Il sera donc utilisable uniquement dans des cas très particuliers.

Indispensables

Une liste d'applications indispensables ou simplement très utiles pour le développement PHP/MySQL.

- **PHP – Linux & Windows – ★ ★ ★ ★ ★**
  L'interpréteur PHP est bien sûr très utile pour développer en PHP ...
- **Apache – Linux & Windows – ★ ★ ★ ★ ★**
  Serveur.
- **PHPMyAdmin – Linux & Windows – ★ ★ ★ ★ ★**
  Interface PHP permettant de gèrer en environnement graphique un base MySQL.
- **EasyPHP – Windows – ★ ★ ★ ★ ★**
  Cette application contient Apache, PHP et MySQL ... le rêve pour ceux qui utilisent un système cauchemardesque ...
- **XamPP – Linux & Windows – ★ ★ ★ ★ ★**
  Equivalent de EasyPHP (contient Apache, MySQL, PHP, Perl, un serveur FTP et PHP-MyAdmin). Disponible bien sûr pour Windows mais aussi pour Linux (là l'utilité est tout de suite nettement moins frappante ...).

Divers

- **KXSLDbg – Linux – ?**
  Débuggeur XSLT (identique au débuggeur de BlueFish Editor et de Quanta Plus).
- **Dia – Linux – ★ ★ ★ ★**
  Outils de conception de diagrammes UML.

Note : Dans la suite, lorsque des indications d'installation ou de configuration seront données, elles le seront toujours par rapport à la distribution Debian Linux.
De plus, s'agissant de la première version du manuscript, il est fort probable que de nombreuses fautes d'orthographe se soient glissées dans ces lignes. Je m'en excuse par avance et si vous me les signalez, je me ferai un plaisir de les corriger ;-)

1

L'architecture client/serveur

Software is like sex :
It's better when it's free.
Linus Torvalds

MATÉRIELLEMENT , un site est constitué d'un ordinateur connecté à internet, et sur lequel tourne en permanence un programme serveur – on le désignera simplement par le terme *serveur*. Le serveur (Apache par exemple) est en attente de requêtes qui lui sont transmises sur le réseau par un programme client[1] tournant sur une autre machine. Lorsque la requête est reçue, le serveur l'analyse afin de déterminer la ou les actions à exécuter, puis transmet la réponse au programme client. Il existe de nombreux types de communication possibles entre serveur et client :

- Le cas le plus simple est une recherche de document. Lorsque l'on se déplace dans un site, en cliquant sur un lien, on effectue une requête pour charger une nouvelle page.
- Le client peut transmettre des informations au serveur et lui demander de les conserver.
- Et enfin, le client peut demander au serveur d'exécuter un programme en fonction de paramètres – cas des CGI –, et de lui transmettre le résultat (exemple `http://www.cmi.univ-mrs.fr/~tristan/bioinfo/gcta/index.html`).

La figure 1.1 illustre les aspects essentiels d'une communication web dite statique. Le client envoie une requête au serveur. Ce dernier se charge de rechercher le document demandé parmi l'ensemble de fichier auxquels il a accès, et transmet ce document s'il existe (sinon vous aurez droit au fameux "404 - page not found").

Le système de localisation de fichiers sur le Web permet de faire référence de manière unique à un document. Il s'agit des URL – Universal Resource Location. Une URL est constituée de plusieurs partie :

[1] La plupart du temps le programme client est un navigateur internet.

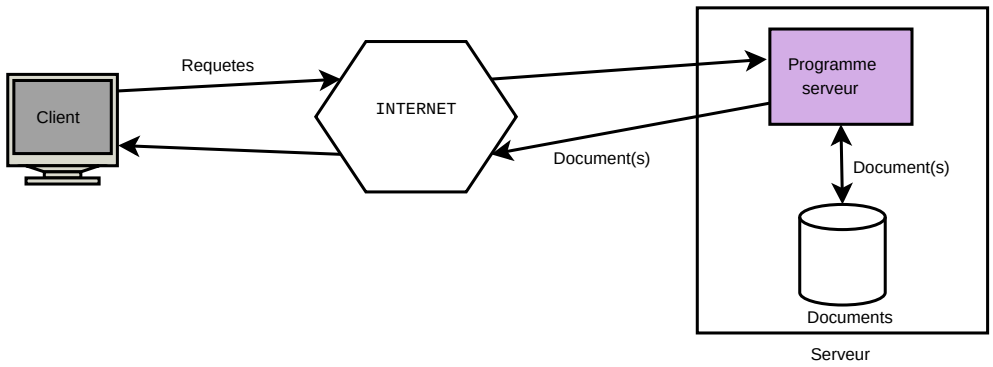


FIG. 1.1 – Architecture Web "classique"

- le nom du protocole utilisé pour accéder à la ressource ; les plus utilisés sont http et ftp ainsi que leurs versions sécurisées shttp et sftp. Le nom du protocole est séparé de la suite de l'URL par les symboles " ://".
- le nom du serveur hébergeant la ressource. Par exemple `lcb.cnrs-mrs.fr`
- le numéro de port réseau sur lequel le serveur est à l'écoute. Il est séparé du nom du serveur par le symbole " :". Par défaut, en http le port est fixé à 80 et en ftp à 21. L'appel de `http://lcb.cnrs-mrs.fr` et de `http://lcb.cnrs-mrs.fr:80` est donc équivalent.
- le chemin d'accès, sur la machine serveur, à la ressource.

Les clients web permettent de dialoguer avec un serveur en lui transmettant ces URL, puis affichent à l'écran les documents transmis par le serveur.
Abordons maintenant les aspects programmation.

1.1 Programmation côté client : JavaScript

La programmation côté client permet de rendre les pages HTML plus réactives sans avoir à effectuer sans cesse des échanges avec le programme serveur. Le JavaScript permet par exemple d'effectuer ce type de programmation. Il est interprété au niveau du client et permet toutes sortes de calculs, d'affichages et de contrôles localement, au niveau du client.
Le JavaScript a été introduit par Netscape puis repris par les autres navigateurs. Il est aujourd'hui plus ou moins normalisé par le W3C, mais il reste toujours difficile de s'assurer qu'un code un tant soit peu complexe donne les mêmes résultats selon le navigateur employé. Les principaux cas de recours à JavaScript sont :

- faire un calcul en local.
- contrôler une zone de saisie (passer tous les caractères d'une zone texte en majuscule par exemple).
- afficher un message.
- enfin piloter l'interface du navigateur (fenêtres, menus déroulants).

Le code JavaScript est inclus dans une page HTML entre des balises `<SCRIPT>` et `</SCRIPT>`. Il peut se trouver dans un fichier séparé. Il suffira alors de le charger dans la page HTML par

un code du type :
`<SCRIPT language="text/javascript" src="fichier.js" />`
On notera ici l'omission de la balise `</SCRIPT>` qui est en fait incluse à la fin de la balise `<SCRIPT ... />`. Ce procédé est valable pour toutes les balises que l'on referme immédiatement après ouverture.
Remarque : Par défaut, Apache va rechercher les fichiers de script dans le répertoire `/var/www` qui est le répertoire des images. Vous pouvez le modifier en éditant le fichier `/etc/apache/httpd.conf` et en modifiant la valeur des lignes :

```
DocumentRoot /var/www
...
<Directory /var/www>
```

Les limites de la programmation côté client sont très vite atteintes : aucune interaction possible avec le serveur.

1.2 Programmation côté serveur : CGI

Le *Common Gateway Interface* (CGI) consiste à produire des documents HTML au travers d'un programme qui est associé au serveur web. Ce programme peut recevoir en outre des paramètres saisis par l'utilisateur dans des formulaires. Les CGI peuvent être programmés dans n'importe quel langage, mais ils le sont en général en Perl ou en C. Voici un exemple très simple de CGI-Perl : *exemple1.cgi*

```
#!/usr/bin/perl -X

use vars qw/$DB $URL \%PAPERS/;
use CGI 2.42 qw/:standard :html3 escape/;
use CGI::Carp qw/fatalsToBrowser/;

$a=param(a);
$b=param(b);
print "
<HTML>

<HEAD>
<TITLE>Programme de test CGI</TITLE>
</HEAD>

<BODY background=\"white\">
Le résultat de ".$a." x ".$b." est : ".$a*$b." <BR />
</BODY>

</HTML>\n";
```

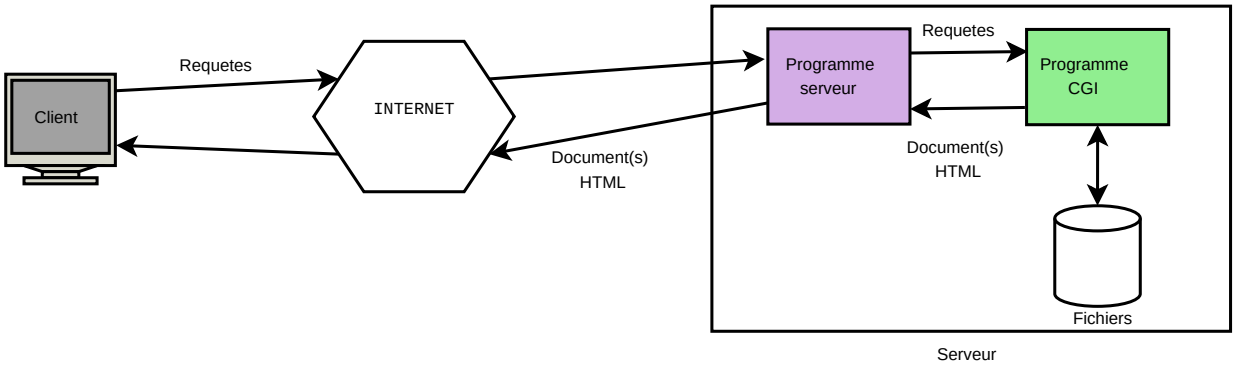


FIG. 1.2 – Architecture CGI

`$a` et `$b` sont des paramètres récupérés depuis un formulaire appartenant à une page HTML et ou les entrées seraient nommées `a` et `b`. On pourrait ainsi avoir :

```
...
<FORM action="exemple1.cgi" method="POST">
Entrez la valeur de a : <INPUT type="text" name="a" size="3" /><BR />
Entrez la valeur de b : <INPUT type="text" name="b" size="3" /><BR />
<INPUT type="submit" name="submit_1" value="Multiplier !" />
</FORM>
...
```

Il n'y a bien sûr ici aucun test d'erreur ... mais il faut penser que l'utilisateur ne va pas forcément donner deux entiers ! Il existe deux méthodes pour transmettre des données au serveur : la méthode GET et la méthode POST. Nous y reviendrons plus tard.

Le CGI est la solution la plus ancienne, et sans doute encore la plus utilisée, pour la gestion de sites web dynamiques. La figure 1.2 illustre les composants d'une architecture CGI. Le client envoie une requête (souvent à partir d'un formulaire HTML, ce qui permet de transmettre des variables) qui est plus complexe que la simple demande de transmission d'un document. Cette requête consiste à faire déclencher une action (déterminée par le champ *action* de la balise `<FORM>`) sur le serveur. Les CGI peuvent par exemple accéder à une base de données pour fournir des résultats.

L'exécution du programme CGI par le serveur web se déroule en trois phases :

- *Requête du client au serveur* : le serveur récupère les informations transmises par le client (le navigateur), c'est-à-dire le nom du programme CGI accompagné, le plus souvent, de paramètres saisis par l'utilisateur dans un formulaire.
- *Exécution du programme CGI* : le serveur déclenche l'exécution du programme CGI en lui fournissant les paramètres reçus.
- *Transmission du document HTML* : le programme CGI renvoie le résultat de son exécution

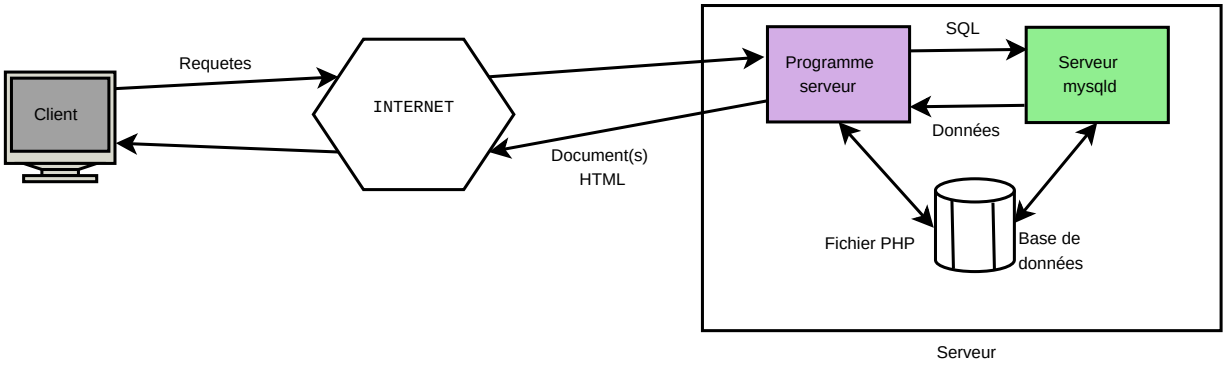


FIG. 1.3 – Architecture PHP/MySQL

au serveur sous la forme d'un fichier HTML, le serveur se contentant alors de faire suivre au client.

1.3 Programmation côté serveur : PHP/MySQL

Une autre solution de programmation côté serveur est le PHP, qui est très souvent associé à MySQL. Un script PHP est exécuté par un interpréteur généralement intégré à Apache sous forme de module. Quand un fichier PHP est demandé au serveur (le client solicite l'affichage d'une page d'extension .php[2]), ce dernier le charge en mémoire et transmet le script PHP à l'interpréteur. Celui-ci exécute le script et produit du code HTML qui vient remplacer le script PHP dans le document transmis au client. Ce dernier ne reçoit donc que du code HTML et ne voit jamais la moindre instruction PHP. Le principe est très proche des CGI avec des améliorations notables :

- Il s'agit d'un langage enfoui (*embeded*) : à l'intérieur d'un code HTML on va inclure des instructions PHP. L'intégration se fait dnc de manière très souple.
- Les variables issues des formulaires sont fournies directement et sans effort.
- Enfin, les scripts sont effectués directement au sein d'Apache, ce qui évite d'avoir à lancer systématiquement un programme CGI.

Un des grands atouts de PHP est sa possibilité de se connecter à un serveur MySQL (démon *mysqld*) pour récupérer est exploiter des données. L'architecture de ce système est donnée en figure 1.3. Il s'agit d'une architecture à trois composantes, chacune réalisant une deds trois tâches fondamentales d'une application.

- Le navigateur constitue l'*interface graphique* dont le rôle est de permettre à l'utilisateur de visualiser et d'interagir avec l'information.
- MySQL est le *serveur ded données*.
- Enfin l'ensemble des fichiers PHP contenant le code d'extraction, traitement et mise en

[2]L'extension des scripts PHP est paramétrable dans le fichier de configuration d'Apache : `/etc/apache/httpd.conf`.

forme des données constitue le *serveur d'application*, associé à Apache qui se charge de transférer les documents produits sur internet.

Dans l'exemple présenté, le serveur mysqld et Apache se situent sur la même machine mais d'un point de vue technique il est aussi simple de les séparer physiquement.

Nous allons maintenant nous intéresser à la programation en PHP proprement dite.

2

PHP

There's More Than One Way To Do It – slogan de Perl.
Larry Wall

Le langage PHP découle d'un autre langage du même auteur, Rasmus Lerdorf. C'est en 1995 qu'il créa PHP/FI qui était initialement une collection de scripts Perl. La collection de scripts originellement appelée "*Personnal Home Page Tools*" a ensuite évolué et a été réimplémentée en C. Mais la syntaxe est restée très proche du Perl, et pour ceux qui connaissent ce langage, les similitudes seront évidentes. PHP rencontrant de plus en plus d'adeptes a pu se développer et acquérir de nouvelles fonctionnalités au cours des versions. Nous en sommes actuellement à la version 5 (PHP 5) dans laquelle l'avancée la plus significative est l'amélioration du modèle objet.

2.1 Les Bases du langage

Dans cette partie nous allons détailler les bases du PHP, c'est-à-dire la syntaxe générale et l'utilisation des variables.

2.1.1 Enfouissement dans du code HTML

Les instructions PHP sont enfouies dans du code HTML. Il faut donc une balise permettant au serveur HTTP de détecter les zones de code. Cette balise est notée `<?PHP ... ?>`. Toute instruction se trouvant dans ce bloc sera interprétée et fournira du code HTML qui sera inséré à la place du bloc pour être ensuite envoyé au client.
Remarque : En utilisant la configuration par défaut d'Apache (fichier `/etc/apache/httpd.cnf`), vos fichiers devront porter l'extension `.php` et devront être placés dans `/var/www` pour être interprétés.

Pour vérifier que PHP est correctement installé sur votre système, il existe une fonction `phpinfo()` permettant d'afficher toutes les informations concernant la configuration de PHP. Cette fonction est destinée à afficher directement une page d'information, elle gère donc elle-même l'ajout des balises `<HTML>...</HTML>`. Il suffit alors de créer le script suivant : *info.php*

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

Dans un navigateur, en tapant l'adresse `http://localhost/info.php`, on obtient alors l'écran présenté en figure 2.4

2.1.2 Les variables

Tout comme le Perl, le PHP n'est pas un langage "typé" obligeant à définir le type de la variable et par là même le type unique de données qu'elle peut contenir. De plus, l'initialisation des variables se fait de manière automatique en fonction du contexte. Une variable non initialisée dans un contexte scalaire (addition par exemple) aura pour valeur 0. Ce procédé est très pratique lorsque l'on sait ce que l'on fait ... mais peut conduire à des bugs difficilement identifiables ! En effet, si vous utilisez une variable `$Variable` dans laquelle vous faites des calculs puis que vous y faites référence en tant que `$Varaible` (si,si, ça peut arriver ...), votre code sera syntaxiquement correct mais sémantiquement faux (il ne fera pas ce que vous attendiez de lui). Vous voilà donc prévenus.
Il existe trois sortes de variables. Tout nom de variable commence par le symbole `$`.

2.1.2.1 Les scalaires

Une variable scalaire peut contenir indifféremment un entier, un réel, du texte, une lettre, etc. Ainsi, pour affecter la valeur 10 à la variable `$var1` et 2 à la variable `$var2$`, nous ferons :

```
$var1=10;
$var2=2;
```

Pour affecter une chaîne de caractères à la variable `$var3`, nous ferons de même :

```
$var3="coucou";
```

Conversion & Typage
Attention, l'addition entre toutes ces variables est permise et ne provoque aucune erreur. En effet, dans un contexte entier, la variable `$var3` contenant pourtant la chaîne `"coucou"` vaut 0. Ainsi, avec :

```
$total=$var1+$var2;
```

La variable `$total` vaudra bien 12, mais avec :

```
$total=$var1+$var3;
```

La variable `$total` vaudra $10 + 0$ soit 10.
Plus vicieusement, si l'on possède la variable `$var4="3 petits cochons"`, alors avec :

```
$total=$var1+$var4;
```

La variable `$total` ne vaudra non pas 10 mais 13 car *var*4 aura été converti en 3. Dans ce cas de conversion la règle est simple : PHP crée un nombre à partir de tous les chiffres qu'il va

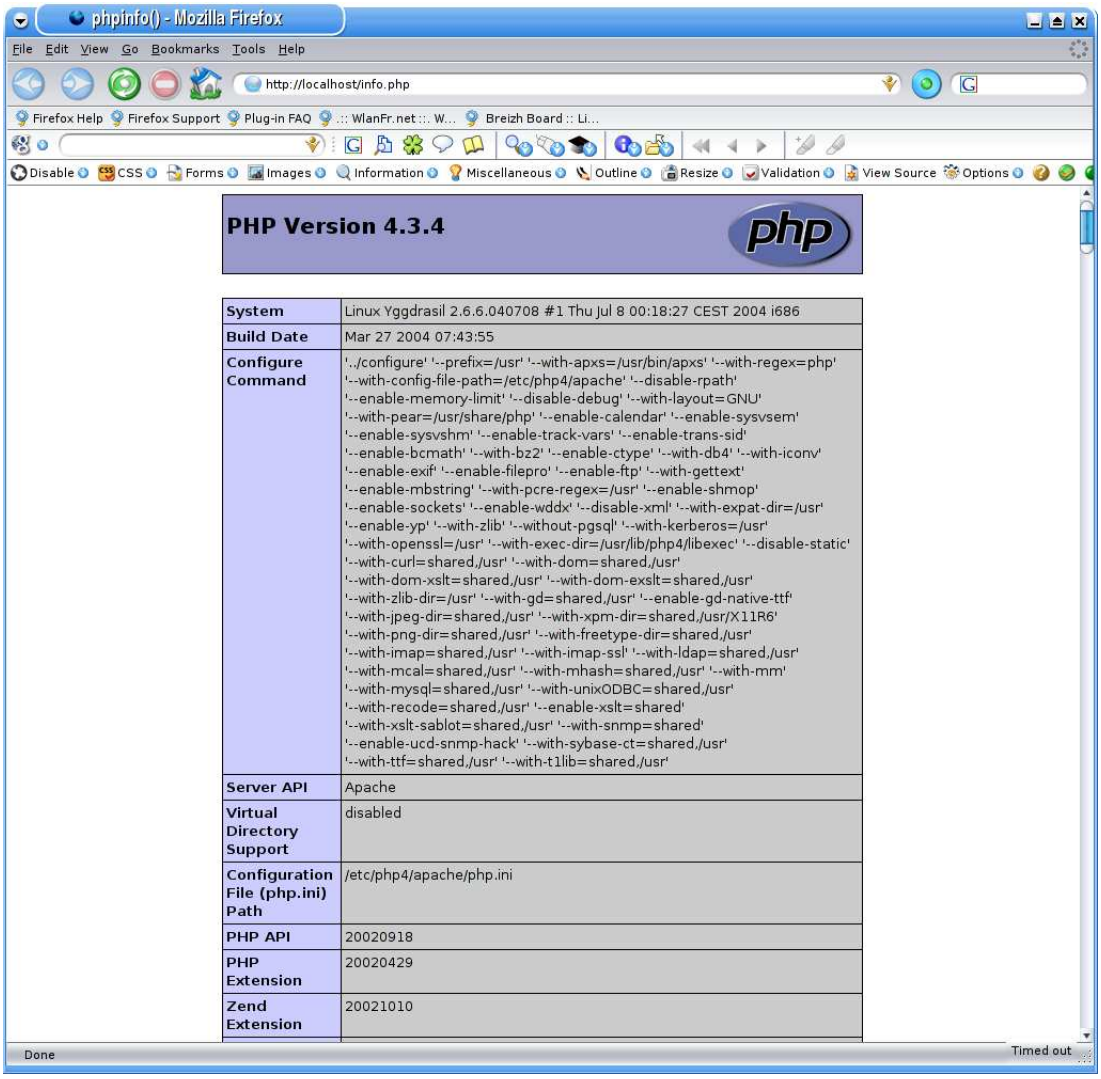


FIG. 2.4 – Aperçu de la page obtenue en utilisant la fonction phpinfo().

rencontrer consécutivement à partir du début de la chaîne (`"a2b5"` converti en entier donnera par exemple 0 mais `"25rtf"` donnera 25).
Pour une conversion explicite, les opérateurs de "cast" sont disponibles et suivent la même syntaxe qu'en Perl (ou en C).
```
$var5=(integer) $var4;
```

Variable de variable
Le nom d'une variable peut lui-même être une variable.
```
$Nom_Var="i";
$$Nom_Var=10;
```
Nous venons de créer une variable `$i` ayant pour valeur `10`. Les applications sont multiples ... mais à utiliser avec beaucoup de précautions.

Chaînes de caractères
Les chaînes de caractères peuvent être encadrées par des guillemets simples (') ou des guillemets doubles ("). Les règles sont les mêmes que dans un shell Linux : les expressions entre guillemets doubles seront évaluées, pas celles entre guillemets simples.
```
$chaine1="$var1 et $var2";
$chaine2='$var1 et $var2';
```
La variable `$chaine1` contient `"10 et 2"` et la variable `$chaine2` contient `"$var1 et $var2"`.

Les constantes
Les constantes sont simplement définies par `define()`.
```
define("PI", "3.14");
```
Le mot-clé `const` est quant à lui utilisé pour déclarer un paramètre de fonction en tant que constante.

Incrémentations
Bien entendu, les opérateurs d'incrémentation (`++`) et de décrémentation (`--`) sont disponibles :
```
$var1++;
--$var2;
```
La variable `$var1` vaut maintenant 11 et la variable `$var2` vaut 1.

Opérateurs
Les opérateurs se répartissent en cinq types : les opérateurs arithmétiques, les opérateurs de bits, les opérateurs logiques, les opérateurs de comparaison et enfin les opérateurs de concaténation de chaînes de caractères.

Opérateurs arithmétiques
Ce sont les opérateurs classiques avec l'addition (`+`), la soustraction (`-`), la multiplication (`*`), la division (`/`) et le modulo (`%`) ou reste de la division entière.

Opérateurs de bits
Là encore, ce sont les opérateurs classiques : le ET binaire (`&`), le OU binaire (`|`), le OU EX-

CLUSIF – XOR – binaire (^), et les décalages de n bits vers la gauche ou la droite (<< et >>).

Opérateurs logiques
Les opérateurs logiques renvoient une valeur interprétée à vrai si elle est différente de 0 ou de la chaîne vide et fausse sinon. Il y a le ET (&& ou and), le OU (|| ou or), le XOR (xor), et le NOT (!).

Opérateurs de comparaison
Ici encore, ce sont les opérateurs classiques : le test d'égalité (==), le test de différence (!=), le test d'infériorité (<), inférieur ou égal (<=), et le test de supériorité (>), supérieur ou égal (>=).

Concaténations de chaînes
L'opérateur de concaténation de chaînes est le point "." :
```
$chaine1="salut";
$chaine2=" a toi";
$concat1=$chaine1.$chaine2;
$concat2=$chaine1;
$concat2.=$chaine2;
```
Les valeurs résultantes de concaténation pour les variables `$concat1` et `$concat2` sont équivalentes. Elles contiennent toutes deux `"salut a toi"`.

2.1.2.2 Les tableaux

Un tableau est une suite de valeurs[3] référencées par une unique variable. PHP gère dynamiquement la taille des tableaux ce qui permet d'ajouter ou de supprimer des éléments à volonté sans se soucier de l'espace nécessaire ni de la déclaration.
De plus, PHP gère l'affectation automatique d'un indice à un nouvel élément du tableau. Cet indice est le numéro de la première cellule vide.
```
$tab[0]=1;
$tab[1]=2;
$tab[3]="trois";
```
Le tableau `$tab` contient maintenant trois éléments. En utilisant l'affectation automtique, pour le même résultat, on aurait tapé :
```
$tab[]=1;
$tab[]=2;
$tab[]="trois";
```
Les affectations se font alors automatiquement de `$tab[0]` à `$tab[1]`.
L'instruction `array()` permet d'initialiser facilement un tableau. Pour obtenir le même résultat que précédemment, on pourrait taper :
```
$tab=array(1, 2, "trois");
```

[3]Ces valeurs peuvent être de différents types puisque PHP n'est pas un langage "typé".

En ce qui concerne les tableaux multi-dimensionnels, il suffit de multiplier les références entre crochets : `$tab[0][0]$` désigne par exemple la première cellule d'un tableau à deux dimensions. Nous verrons par la suite que l'affichage du contenu de ces variables multi-dimensionnelles ne peut pas être effectué de manière classique.

2.1.2.3 Les tables de hachage

Les tables de hachages ressemblent beaucoup aux tableaux, si ce n'est que l'on fait référence à une cellule à l'aide d'un mot-clé – appelé *clé* – au lieu d'utiliser un entier. Cette structure est vraiment très utile. Par exemple :

```
$nom{"Ross"}="Geller";
$nom{"Chandler"}="Bing";
$nom{"Joey"}="Tribbiani"
```

On peut ainsier obtenir très facilement le nom d'un personnage en fonction de son prénom. Comme avec les tableaux, on peut utiliser l'instruction `array()` lors de l'initialisation :

```
$nom=array(
 "Ross" => "Geller",
 "Chandler" => "Bing",
 "Joey" => "Tribbiani"
);
```

On peut créer également des tables de hachage multi-dimensionnelles :

```
$metier{"Geller"}{"Ross"}="Dinosaures";
$metier{"Geller"}{"Monica"}="Chef cuisinier";
$metier{"Green"}{"Rachel"}="Styliste";
```

Avec une telle table nous pouvons par exemple connaître le métier exercé par tous les personnages ayant le même nom "Geller".
Nous verrons par la suite comment parcourir les tables de hachage (cf Instructions de boucles – foreach).

2.1.2.4 Les tableaux mixés

Les tableaux multi-dimensionnels peuvent être des tableaux mixés si l'on associe les tables de hachage et les tableaux. Par exemple :

```
$enfant{"Anakin"}[0]="Luke";
$enfant{"Anakin"}[1]="Leïa";
```

Ce qui équivaut à :

```
$enfant=array(
 "Anakin" => array("Luke", "Leïa")
);
```

2.1.2.5 Relation entre tableaux et chaînes de caractères

Comme dans la plupart des langages évolués, une chaîne de caractères est en fait un tableau où chaque case contient une lettre :
`$chaine="Ross Geller";`
La variable `$chaine[0]` contient "`R`", `$chaine[1]` contient "`o`", ...

2.1.3 Les variables d'environnement

Les variables d'environnement sont des données stockées dans des variables permettant au programme d'avoir des informations sur son environnement d'exécution. Dans le cas des scripts PHP, il y a un environnement côté client, et un environnement côté serveur (des variables sont alors créées par le serveur à chaque fois qu'un script PHP est appelé : le serveur lui fournit les variables d'environnement en paramètres cachés lors de l'exécution de l'interpréteur).
Ces variables permettent notamment d'avoir des informations sur le type de serveur, son administrateur, la date à laquelle le script a été appelé, l'adresse IP et le type de navigateur du client,...

2.1.3.1 Les variables d'environnement dépendant du client

Variables d'environnement	Description
`$AUTH_TYPE`	Méthode d'authentification utilisée par le client.
`$CONTENT_TYPE`	Type MIME des données.
`$DOCUMENT_ROOT`	Racine des documents sur le serveur.
`$DOCUMENT_URI`	Chemin relatif du script PHP.
`$HTTP_ACCEPT`	Types MIME reconnus par le serveur (séparés par des virgules).
`$HTTP_ACCEPT_ENCODING`	Types d'encodage que le serveur peut réaliser (gzip,deflate).
`$HTTP_ACCEPT_LANGUAGE`	Langue utilisée par le serveur (par défaut en-us).
`$HTTP_CONNECTION`	Type de connexion ouverte entre le client et le serveur.
`$HTTP_HOST`	Nom d'hôte de la machine du client associée à l'adresse IP.
`$HTTP_REFERER`	URL de la page qui a appelé le script PHP.
`$HTTP_USER_AGENT`	Type de navigateur/système d'exploitation utilisés par le client.
`$LAST_MODIFIED`	Date et heure de dernière modification du fichier.
`$PATH`	Chemin d'accès aux différents répertoires sur le serveur.
`$PATH_INFO`	Chemin relatif d'accès au script PHP.
`$PHP_SELF`	Nom du script PHP.
`$REDIRECT_STATUS`	Etat de la redirection (echec ou succès).
`$REDIRECT_URL`	URL vers laquelle le navigateur du client a été redirigé.
`$QUERY_STRING`	Données d'un formulaire transmis par GET (URL après " ?").
`$REMOTE_ADDR`	Adresse IP du client appelant le script CGI.
`$REMOTE_PORT`	Port sur lequel la requête HTTP a été envoyée au serveur.
`$SCRIPT_FILENAME`	Chemin d'accès absolu au script PHP.
`$SCRIPT_NAME`	Chemin d'accès relatif au script PHP.

2.1.3.2 Les variables d'environnement dépendant du serveur

Variables d'environnement	Description
$DATE_GMT	Date actuelle au format GMT.
$DATE_LOCAL	Date actuelle au format local.
$DOCUMENT_ROOT	Racine des documents Web sur le serveur.
$GATEWAY_INTERFACE	Version des spécifications CGI utilisées par le serveur.
$HTTP_HOST	Nom de domaine du serveur.
$SERVER_ADDR	Adresse IP du serveur.
$SERVER_ADMIN	Adresse de l'administrateur du serveur.
$SERVER_NAME	Nom donné au serveur en local.
$SERVER_PORT	Numéro de port associé au protocole HTTP sur le serveur.
$SERVER_PROTOCOL	Nom/version du protocole utilisé pour envoyer la requête.
$SERVER_SOFTWARE	Type (logiciel) du serveur web (ex Apache).

Pour afficher la valeur d'une variable d'environnement, on utilise la fonction `getenv()`. Par exemple :

```
$lang=getenv("HTTP_ACCEPT_LANGUAGE")
```

Pour créer une nouvelle variable d'environnement, c'est `putenv()` :

```
putenv("MA_VARIABLE=10");
```

2.2 Structures de contrôle

Les structures de contrôle permettent d'effectuer des tests et des boucles, et donc d'indiquer l'ordre dans lequel les instructions doivent être traitées. Ces instructions utilisent les opérateurs de comparaison vus précédemment.

2.2.1 Les instructions conditionnelles

Ces instructions permettent d'imposer une condition, et en fonction du fait qu'elle soit vérifiée (vrai) ou pas, de déterminer le code à exécuter. On distingue le test simple et le test "par cas".

2.2.1.1 Si

Il s'agit du test simple, reprenant la syntaxe du C ou du Perl[4]. Si la condition est vérifiée, on exécute un bloc de commandes, sinon on en exécute un autre.

```
if ($meteo=="soleil")
 print "Yeepeeee !";
else
 print "Bouuuh ...";
```

[4] A la différence près qu'en Perl les accolades sont **toujours** obligatoires, même si l'action résultant du test ne contient qu'une seule instruction. Le PHP obéit ici aux règles du C.

A signaler, l'existence d'une syntaxe esotérique dont je déconseille l'utilisation (mais qui existe ...) :

```
if (expression):
 ... bloc de commandes ...
else:
 ... bloc de commandes ...
endif;
```

Ceci permet seulement de se passer des accolades lors de l'écriture des blocs ...

2.2.1.2 Choix

Structure "Switch" empruntée au langage C.

```
switch ($meteo)
{
 case "soleil" : print "Yeepeeee !";
                 break;
 case "pluie"  :
 case "vent"   : print "Bouuuh ...";
                 break;
 default       : print "Il neige ???";
}
```

Remarque : ne pas oublier l'instruction *break* à la fin du traitement d'un cas sinon les instructions suivantes seront exécutée. Dans l'exemple précédent, si `$meteo` contient `"pluie"` alors on se branche sur une ligne vide, puis on exécute le code destiné au cas `"vent"`.

2.2.2 Les boucles

Les instructions de boucle ont été empruntées au Perl (qui en avait lui même emprunté certaines au C).

2.2.2.1 Tant que

Exécute un bloc d'instructions tant que la condition est vraie.

```
while (condition)
{
 ... bloc de commandes ...
}
```

Là encore, vous pourrez rencontrer une notation esotérique :

```
while (condition):
 ... bloc de commandes ...
endwhile;
```

Rappel : il faut bien sûr que la condition varie à l'intérieur du bloc de commande sous peine de créer une boucle infinie.

2.2.2.2 Faire tant que

Variant du `while` qui effectue au moins une fois le bloc de commande puisque le test est effectué après.

```
do
{
 ... bloc de commandes ...
} while (condition);
```

2.2.2.3 Pour

Instruction `for` classique : on spécifie l'initialisation des variables conditionnant l'itération, la condition d'arrêt de la boucle, et l'évolution des variables de boucle. Sur un exemple très simple :

```
for ($i=0; $i<10; $i++)
 print "Etape $i <BR />\n";
```

Affiche "Etape 0" jusqu'à "Etape 9".

2.2.2.4 Pour chaque

L'instruction `foreach` est une instruction particulièrement intéressante issue de Perl[5] et qui permet de parcourir facilement les tableaux (en temps machine cette instruction est plus rapide qu'un `for`). Elle récupère successivement la valeur de chaque cellule. Sur un tableau simple, par exemple, le parcours se fera par :

```
$tab[0]=1;
$tab[1]=2;
$tab[2]=3;
foreach ($tab as $elt)
 print "Element : $elt<BR />\n";
```

La variable `$elt` va successivement prendre les valeurs `$tab[0]` à `$tab[2]`. Dans le cas d'un tableau multi-dimensionnel, il faudra ajouter autant de boucle `foreach` que de dimensions.

```
$tab[0][0]="A1";
$tab[0][1]="A2";
$tab[1][0]="B1";
$tab[1][1]="B2";
foreach ($tab as $tab_niveau1)
 foreach ($tab_niveau1 as $elt)
   print "Element : $elt<BR />\n";
```

La variable `$tab_niveau1` fait référence à `$tab[0]` puis à `$tab[1]`, il faut donc appliquer une autre boucle `foreach` sur ces sous-tableaux pour accéder aux instructions.

[5] La syntaxe a toutefois était modifiée par rapport au Perl où, pour un tableau @tab, l'appel se fait par foreach $elt (@tab).

Dans le cas des tables de hachage, lors du parcours de la table on récupère la clé et la valeur de la cellule.

```
$nom{"Rachel"}="Green";
$nom{"Monica"}="Geller";
$nom{"Phoebe"}="Buffay"
foreach ($nom as $cle => $valeur)
 print "Cle : $cle et Valeur : $valeur<BR />\n";
```

Les clés récupérées ici seront le prénom des personnages (ex. "Rachel") et la valeur contient le nom (ex. "Green"). Dans le cas des tables de hachages multi-dimensionnelles il faudra utiliser plusieurs instructions `foreach` emboîtées.

2.2.3 Les instructions break et continue

Les instructions `break` et `continue` sont des expressions un peu particulières : elles permettent d'interrompre le déroulement d'une boucle et cela même si la condition de boucle est toujours valide.
L'instruction `break` déclenche la sortie forcée de la boucle.
L'instruction `continue` va directement au test de boucle sans exécuter les instructions suivant le `continue`.

```
for ($i=0; $i<10; $i++)
{
 if ($i==3) continue;
 if ($i==7) break;
 print "Etape $i<BR />\n";
}
```

Cette boucle affiche les étapes 0 à 2 (étape 3 sautée par le `continue`) puis 4 à 6 (sortie avant l'impression de l'étape 7 par le `break`).

2.3 Les commentaires

Des commentaires peuvent être insérés dans le code. Tous les symboles suivant `//` ou compris entre `/*` et `*/` ne seront pas interprétés.

2.4 Les fonctions

Une fonction est une instruction permettant de regrouper l'exécution d'un groupe d'instructions dépendant éventuellement de paramètres. En PHP, les fonctions doivent être définies avant leur appel. La déclaration d'une fonction obéit aux règles du C exceptés pour les arguments qui ne sont bien sûr pas typés. La déclaration typique d'une fonction se fait par :

```
function NomDeLaFonction ([$arg1, ..., $argn])
{
 ... instructions ...
}
```

Une fonction ne peut renvoyer qu'une seule valeur par l'intermédiaire de l'instruction `return`, mais cette valeur peut être un tableau ...

```
function CreateTable()
{
 $tab[0]=1;
 $tab[1]=2;
 return($tab);
}

$table=CreateTable();
```

2.4.1 Arguments

Les arguments d'une fonction peuvent être passés par valeur – une copie des variables est faite au moment de l'appel de la fonction, ou par adresse – la fonction travaille directement sur les variables qui lui sont passées en argument. Prenons l'exemple d'une fonction additionnant deux entiers :

```
function addition($a, $b)
{
 $res=$a--+$b;
 return($res);
}
```

L'appel se fera par : `$n1=2;`

```
$n2=5;
$sum=addition($n1, $n2);
```

En sortie, on aura `$sum=7`, `$n1=2`, et `$n2=5`. Mais attention ! Même si la fonction a été conçue pour travailler sur des variables passées par valeur, il est possible de passer les arguments par adresse grâce au symbole `&`.

```
$sum=addition(&$n1, $n2);
```

En sortie `$sum=7`, `$n2=5`, mais `$n1=1` ! D'autre part, il est possible de déclarer une fonction prenant des arguments passés par adresse :

```
function addition(&$a, $b)
{
 $res=$a+$b;
 $a=10;
 return($res);
}
```

Dans ce cas, en effectuant un appel de la fonction de la même manière que précédemment :

```
$n1=2;
$n2=5;
$sum=addition($n1, $n2);
```

Cela produira en sortie `$sum=7`, `$n2=5`, et `$n1=2`. Mais si l'on n'a pas accès à la définition de la

fonction, rien n'indique que le premier argument est passé par adresse ! Le passage d'arguments par adresse est donc a manipuler avec beaucoup de précautions (et même à éviter si possible : on peut tout à fait tout écrire sans recourir au passage par adresse).

2.4.1.1 Valeurs par défaut

Il est possible de définir des valeurs par défaut pour un ou plusieurs arguments d'une fonction. Si la variable est omise lors de l'appel à la fonction, ce sera la valeur par défaut qui sera utilisée.

```
function addition($a, $b=10)
{
 return($a+$b);
}

$n1=2;
$n2=5;
$sum1=addition($n1);
$sum2=addition($n1, $n2);
```

`$sum1` contiendra 12 $(2 + 10)$ et `$sum2` contiendra 7 $(2 + 5)$.

2.4.1.2 Contrôle du nombre d'arguments

Si le nombre d'arguments fournis à une fonction diffère du nombre d'arguments déterminés dans la déclaration de la fonction, PHP exécute quand même cette fonction. S'il y a plus d'arguments, les arguments en trop sont ignorés. S'il n'y a pas assez d'arguments, un message d'avertissement – *warning* – sera affiché, et la fonction sera exécutée en utilisant des variables déclarées par défaut. Par exemple, dans un contexte de calcul entier la variable vaudra 0.

```
function addition($a, $b)
{
 return($a+$b);
}

$n1=2;
$n2=5;
print "$n1+$n2=".addition($n1)."<BR />\n";
```

L'affichage produit sera : `"2+5=2"` puisque la variable `$b` de la fonction addition n'ayant pas été passé en argument, PHP a créé et utilisé une variable de valeur 0.
Pour remédier à ce problème (dans le cadre de la création d'une librairie par exemple), on peut tester dans la fonction si tous les arguments sont bien présents.

```
function addition($a, $b)
{
 if (!$a || !$b)
```

```
 {
   print "Il manque un argument !<BR />\n";
   exit;
 }
 else
   return($a+$b);
}
```

Comme les variables manquantes sont initialisées par défaut à des valeurs booléennes fausses (0 ou `""`), le test porte justement sur ces valeurs. Une alternative au test `if (!$a || !$b)` est d'utiliser la fonction `empty()` qui renvoie vrai si la variable passée en argument n'a pas été initialisée (ou vaut 0 ou `""`) : `if (empty($a) || empty($b))`. A noter que si l'une des deux variables vaut 0 par choix de l'utilisateur, on affichera aussi le message d'erreur. Là encore, il faut être très prudent dans la nature des tests effectués.

2.4.2 Portée des variables

Il existe trois types de portée de variables.

- *Variables automatiques.* Ces variables sont crées lors de leur définition, et restent valables dans leur espace de définition (script, fonction, ...). Elles disparaissent lorsque l'on quitte cet espace. Par exemple, une variable déclarée à l'intérieur d'une fonction ne pourra être utilisée à l'extérieur de cette fonction.
- *Variables statiques.* Ces variables sont déclarées à l'intérieur d'une fonction et ne sont pas visibles de l'extérieur, mais entre deux appels à la fonction, la variable statique conserve sa valeur. Par exemple :

  ```
  function CompteAppels()
  {
   static $cpt=0;
   $cpt++;
   print "Passage n°$cpt<BR />\n";
  }
  ```

  Lors du premier appel à `CompteAppels()` on obtiendra `"Passage n°1"`, puis au second `"Passage n°2"` et ainsi de suite.
- *Variables globales.* Une variable globale est visible en tout endroit du programme (même si elle est déclarée à l'intérieur d'une fonction).

  ```
  function BeArK()
  {
   global $var="beark !";
  }

  ... instructions ...
  $var="variable";
  BeArK();
  print "var=$variable";
  ```

Après exécution de ces instructions, la variable `$var` contient `"beark !"`. L'utilisation de ces variables est très fortement déconseillée (manque de lisibilité du code, manque de sécurité, et enfin manque d'évolutivité).

2.5 Les expressions régulières

Suivant les fonctions de manipulation de chaînes de caractères que vous utiliserez, vous aurez besoin des expressions régulières. En voici un rapide rappel :

Définition	**Caractère E.R.**
0...9	\d
a...zA...Z0...9_	\w
\t \n \r \f	\s
suite de caractères de x à y	[x-y]
début de ligne	^
fin de ligne	$
n'importe quel caractère	.

On peut appliquer à ces définitions les opérateurs suivants :

Opérateur	**Caractère E.R.**
répétition 0 à n fois	*
répétition 1 à n fois	+
répétition x fois	{x}
répétition au moins x fois	{x,}
répétition entre x et y fois	{x, y}
un caractère parmi une liste	[ ]

Pour détecter un identificateur (une lettre minuscule ou majuscule suivie de lettres minuscules ou majuscules, de chiffres ou de caractères "_"), on pourra par exemple écrire : `/[a-zA-Z]\w*/`. Si l'on veut les identificateurs qui contiennent au moins trois chiffres consécutifs : `/[a-zA-Z]\w*\d{3,}\w*/`.

2.6 Quelques fonctions PHP

Voici maintenant quelques fonctions PHP essentielles. Elles sont présentées par groupe d'intérêt avec un exemple d'application pour les fonctions les plus délicates à utiliser. Vous trouverez sur le site `http://www.php.net` les descriptions de toutes les fonctions PHP généralement accompagnées d'exemples d'utilisation.
Les arguments donnés entre crochets sont optionnels.

2.6.1 Fonctions de base

2.6.1.1 echo "*string* Chaine"

Affiche la chaîne de caractères *Chaine*.

2.6.1.2 *boolean* empty(*variable*)

Renvoie vrai si la variable est indéfinie ou si elle a une valeur nulle (`""` ou 0).

2.6.1.3 *string* exec(string *commande* [, string *tableau* [, int *retour*]])

Exécute une commande système et renvoie la dernière ligne produite par la-dite commande. Le paramètre optionnel *tableau* contient toutes les lignes produites par l'exécution de la commande et l'autre paramètre optionnel *retour* renvoie la valeur de retour de la commande.

```
exec("ls", $tab); foreach ($tab as $ligne)
 print "$ligne<BR />\n";
```

2.6.1.4 *boolean* define(*string* Nom, *mixed* Valeur [, *boolean* Insensible])

Définit une constante *Nom*. Si *Insensible* est vrai, alors le nom de la constante est insensible à la casse (une variable `toto` pourra être appelée par `Toto`, `ToTO`, ...).

```
define(PI, "3.14");
```

2.6.1.5 *boolean* defined(*string* Nom)

Renvoie vrai si la constante *Nom* est définie.

2.6.1.6 *int* ereg(*string* ER, *string* Chaine [, *array* Occurences])

Recherche l'expression régulière *ER* dans *Chaîne*. Renvoie Vrai si elle est trouvée et faux sinon. Si le tableau *Occurences* est spécifié, place les occurences trouvées dans ce tableau.

2.6.1.7 *string* ereg_replace(*string* ER, *string* Remplacement, *string* Chaine)

Recherche l'expression régulière *ER* dans la chaine de caractères *Chaine* et la remplace par la chaine *Remplacement* si elle est trouvée. La chaîne modifiée est ensuite renvoyée par la fonction.

```
$chaine_initiale="Salut c'est moi!";
$chaine_modifiee=ereg_replace("moi", "toi", $chaine_initiale);
```

2.6.1.8 *boolean* mail(*string* Destinataire, *string* Sujet, *string* Texte, *string* EnTete)

Envoie un e-mail. Voici comment appliquer cette fonction :

```
$to_mail     = $Adresse_mail_du_destinataire;
$from_email  = $Adresse_mail_de _l_expediteur;
$entetedate  = date("D, j M Y H:i:s -0600");
$entetemail  = "From: $from_email \n"; // Adresse expéditeur
$entetemail .= "Cc: \n";
$entetemail .= "Bcc: \n"; // Copies cachées
$entetemail .= "Reply-To: $from_email \n"; // Adresse de retour
$entetemail .= "X-Mailer: PHP/" . phpversion() . "\n" ;
$entetemail .= "Date: $entetedate";
$sujet="Sujet du mail";
$texte_du_mail="Texte de votre mail";
mail($to_mail, $sujet, $texte_du_mail, $entetemail);
```

On peut également envoyer des mails grâce au paquetage PEAR `MAIL_mime`. Ce paquetage permet également de transmettre des pièces jointes. Voici un exemple d'utilisation :

```
require("Mail.php");
require("Mail/mime.php");

$mail=new Mail_mime;

$headers{"From"}="expediteur@toto.com";
$headers{"Subject"}="Sujet du Mail";
$a="destinataire1@toto.com, destinataire2@toto.com";

$texte="Texte du mail (version texte)";
$mail->setTXTBody($texte);

$html="<HTML><BODY>Version HTML du mail</BODY></HTML>";
$mail->setHTMLBody($html);

$fichier_a_joindre="/repertoire/fichier.gif";
$mail->addAttachment($fichier_a_joindre, "image/gif");

$headers=$mail->headers($headers);
$body=$mail->get();
$message=& Mail::factory("mail");
$message->send($a, $headers, $body);
```

2.6.1.9 print "*string* Chaine"

Voir echo.

2.6.1.10 printf(*string* Format, var_1, ..., var_n)

Fonction `printf()` identique à celle du C.

```
printf("Etiquette n°%d\n", 10);
```

2.6.2 Fonctions de manipulations de chaîne de caractères

2.6.2.1 *string* chop(*string* Chaine)

Renvoie la chaîne *Chaine* après avoir supprimé tous les caractères espace en fin de chaîne.

2.6.2.2 *array* explode(*string* Separateur, *string* Chaine)

Renvoie un tableau dont chaque cellule contient les éléments de *Chaine* découpés au niveau de *Séparateur*.

```
$var="Aragorn;Legolas;Galadriel;Arwen;Eowyn";
$table=explode(";", $var);
```

`$table` est un tableau de 5 cellules.

2.6.2.3 *string* implode(*array* Tableau, *string* Separateur)

Fonction inverse de explode : concatène dans une chaîne de caractères toute les cellules d'un tableau en les séparants par *Separateur*.

```
$table[0]="Minas Tirith";
$table[1]="Minas Morgul";
$table[2]="Dagorlad";
$chaine=implode($table, "/");
```

2.6.2.4 *int* strcmp(*string* Chaine1, *string* Chaine2)

Equivalent de la commande `strcmp()` du C : compare deux chaînes de caractères et renvoie :

- 0 si les deux chaînes sont égales,
- *une valeur négative* si *Chaine1* précède *Chaine2* dans l'ordre lexicographique,
- *une valeur positive* si *Chaine2* précède *Chaine1* dans l'ordre lexicographique.

2.6.2.5 *string* stripSlashes(*string* Chaine)

Renvoie la chaîne *Chaine* après y avoir supprimé les caractères de protection (\) devant (”), (’) et (\).

2.6.2.6 *int* strlen(*string* Chaine)

Equivalent à la fonction `strlen()` du C : renvoie la longueur de *Chaine*.

2.6.2.7 *string* trim(*string* Chaine)

Renvoie *Chaine* après avoir ôté tous les espaces en début et en fin de chaîne.

2.6.3 Fonctions d’accès aux tableaux

2.6.3.1 arsort(*array* Table_Hachage)

Trie *Table_Hachage* sur les valeurs, en ordre descendant.

2.6.3.2 asort(*array* Table_Hachage)

Trie *Table_Hachage* sur les valeurs, en ordre ascendant.

2.6.3.3 *int* count(*array* Tableau)

Renvoie le nombre d’éléments de *Tableau*.

2.6.3.4 *mixed* current(*array* Tableau)

Renvoie la valeur de l’élément courant du tableau (cellule désignée par le pointeur actif). Pour un exemple voir `file()`.

2.6.3.5 *array* each(*array* Table_Hachage)

Renvoie la clé et la valeur de l’élément courant et fait avancer le pointeur d’une cellule.

2.6.3.6 *mixed* key(*array* Table_Hachage)

Renvoie la clé de l’élément courant de *Table_Hachage*.

2.6.3.7 ksort(*array* Table_Hachage)

Trie *Table_Hachage* sur la clé.

2.6.3.8 *mixed* next(*array* Tableau)

Renvoie la valeur de l'élément suivant l'élément courant de *Tableau* et fait avancer le pointeur d'une cellule. Renvoie Faux lorsque le dernier élément est dépassé. Pour un exemple voir `file()`.

2.6.3.9 *mixed* prev(*array* Tableau)

Renvoie la valeur de l'élément précédant l'élément courant de *Tableau* et fait reculer le pointeur d'une cellule. Renvoie Faux lorsque le premier élément est dépassé.

2.6.3.10 reset(*array* Tableau)

Repositionne le pointeur de *Tableau* sur la première cellule.

2.6.3.11 rsort(*array* Tableau)

Trie *Tableau* sur les valeurs, en ordre descendant.

2.6.3.12 sort(*array* Tableau)

Trie *Tableau* sur les valeurs, en ordre ascendant.

2.6.4 Fonctions d'accès aux fichiers

2.6.4.1 fclose(*int* Descripteur)

Ferme le fichier identifié par *Descripteur*. Pour un exemple voir `fopen()`.

2.6.4.2 *boolean* feof(*int* Descripteur)

Renvoie Vrai lorsque la fin du fichier identifié par *Descripteur* est atteinte. Pour un exemple voir `fopen()`.

2.6.4.3 *char* fgetc(*int* Descripteur)

Renvoie le caractère se situant au niveau du pointeur actif et fait avancer le pointeur d'un caractère.

2.6.4.4 *string* fgets(*int* Descripteur, *int* Max_Longueur)

Renvoie la ligne du fichier (de taille maximale *Max_Longueur*) désignée par le pointeur courant et fait avancer le pointeur courant d'une ligne. Pour un exemple voir `fopen()`.

2.6.4.5 *array* file(*string* Nom)

Charge tout le contenu du fichier *Nom* dans un tableau avec une ligne par cellule.

```
$tab_file=file("fichier");
if (empty($tab_file))
{
 print "Impossible d'ouvrir le fichier<BR />\n";
 exit;
}
while ($ligne=current($tab_file))
{
 print "$ligne<BR />\n";
 next($tab_file);
}
```

2.6.4.6 *boolean* file_exists(*string* Nom)

Teste l'existence du fichier *Nom*.

2.6.4.7 *int* filesize(*string* Nom)

Renvoie la taille du fichier *Nom*.

2.6.4.8 *int* fopen(*string* Nom, *char* Mode)

Ouvre le fichier *Nom* et renvoie le descripteur utilisé pour effectuer des opérations sur ce fichier. Trois modes sont disponibles :

- *r* : Lecture seule,
- *w* : Ecriture seule : le contenu du fichier est effacé s'il existe déjà,
- *a* : Ajout : le fichier est créé s'il n'existe pas.

```
$file=fopen("fichier", "r");
if (empty($file))
{
 print "Impossible d'ouvrir fichier";
 exit;
}
while (!feof($file))
```

```
{
 $ligne=fgets($file, 1024);
 print "$ligne<BR />\n";
}
fclose($file);
```

2.6.4.9 *boolean* fputs(*int* Descripteur, *string* Chaine)

Ecrit *Chaine* dans le fichier identifié par *Descripteur*. Renvoie Vrai si l'opération a réussi.

2.6.4.10 *boolean* is_uploaded_file(*string* Nom)

Vérifie l'existence d'un fichier transmis par le client vers le serveur. Le fichier est transmis par un formulaire du type :

```
<FORM enctype="multipart/form-data" action="script.php" method=POST>
 Fichier : <INPUT type="file" size=40 name="MonFichier" /><BR />
 <INPUT type=submit value="Transférer" />
</FORM>
```

Dans `script.php`, le fichier est récupéré dans la variable table de hachage `$_FILES`[6] :

```
if (is_uploaded_file($_FILES{"MonFichier"}{"tmp_name"}))
{
 print "Fichier correctement transféré. On le copie sur le serveur.<BR />\n";
 copy($_FILES{"MonFichier"}{"tmp_name"}, "/data_server/fichier");
}
```

2.6.4.11 *int* readfile(*string* Nom [, *boolean* Chercher_Partout])

Envoie le fichier *Nom* hébergé par le serveur sur le client. Si le paramètre *Chercher_Partout* est spécifié est vaut Vrai, alors le fichier *Nom* est recherché dans toute l'arborescence. Voici par exemple comment transmettre au client le fichier PDF `/data/fichier.pdf` situé sur le serveur :

```
$fichier="fichier.pdf";
$chemin_fichier="/data"/".$fichier;
header("Content-Type: application/pdf");
header("Content-Length: ".filesize($chemin_fichier));
header("Content-disposition: inline; filename=$fichier");
readfile($chemin_fichier);
```

2.7 Les cookies

Un cookie est un fichier texte de taille limitée (65 Ko) permettant de stocker des informations chez le client. On peut ainsi garder en mémoire des informations sur un visiteur de manière à les

[6] Cette table contient tous les fichiers transférés par le client (première clé) et les champs tmp_name (nom du fichier sur le serveur), name (nom du fichier chez le client), size (taille du fichier), et type (type du fichier).

ré-utiliser par la suite (comme son nom, son prénom, ... toute information transmise au serveur). De nombreux utilisateurs n'acceptent pas les cookies sur leur navigateur, ce qui peut engendrer des problèmes d'exécution de votre script s'il nécessite absolument de stocker des informations. On préfèrera l'utilisation des sessions (voir paragraphe ci-après).
Tous les cookies en provenance d'un même site seront stockés dans le même fichier. Un cookie peut alors être assimilé à une variable dont le contenu sera stocké chez le client. Pour créer un cookie on utilise la fonction `setcookie()` qui prend en paramètre le nom du cookie, sa valeur, et la durée de vie du cookie exprimée en secondes (Attention ! L'envoi d'un cookie doit être la première fonction PHP utilisée dans le script : si d'autres fonctions interviennent auparavant, l'envoi du cookie échouera !). Pour définir des cookies permettant de stocker un nom et un prénom pour une durée de un an (365*24*3600 secondes) par exemple, on effectuera :

```
setcookie("Nom", "Geller", time()+365*24*3600);
setcookie("Prenom", "Ross", time()+365*24*3600);
```

Les cookies "Nom" et "Prenom" sont alors stockés sur le client (les cookies ne seront accessibles qu'au chargement de la prochaine page, ou au rechargement de la page courante). Pour pouvoir y accéder depuis un autre script, on utilisera la table de hachage `$_COOKIE`. Par exemple :

```
print "Bonjour, vous êtes ".$_COOKIE{"Prenom"}." ".$_COOKIE{"Nom"}."<BR />\n";
```

Il est également possible de définir des cookies sous forme de tableaux :

```
setcookie("table[0]", "1", time()+365*12*3600);
setcookie("table[1]", "2", time()+365*12*3600);
```

Et pour les utiliser :

```
foreach ($_COOKIE{"table"} as $val)
  print "Elt => $val\n";
```

Enfin, pour effacer un cookie, on le redéclare avec une date d'expiration déjà dépassée (comme ça il est supprimé) :

```
setcookie("Nom", "", time()-3600);
```

Ici par exemple, on a déclaré que la date limite pour ce cookie s'achevait il y a une heure ... donc il est supprimé.

2.8 Les sessions

A partir de PHP4, un mécanisme permettant de passer des informations de page en page pour un utilisateur donné (et de manière sécurisée) a été introduit. Il s'agit des sessions, mécanisme géré de manière transparente par PHP. A chaque fois qu'une session est démarrée, un identifiant unique est attribué au visiteur. Ainsi, pendant la session, on pourra définir à loisir des variables[7] qui "suivront" le visiteur tout au long de sa naviguation. A chaque chargement de page, ces variables seront stockées sous forme sérialisée dans un fichier texte portant le nom de la session (en général dans le répertoire `/tmp`).
La session peut être comparée au cookie mais est plus sécurisée : en effet, elle est stockée sur le serveur et elle est plus difficile d'accès pour les éventuels pirates. Attention, la session est plus sécurisée que le cookie, mais pas inviolable : évitez de stocker des données très sensibles. Enfin,

[7] Ces variables sont de tous types : aussi bien des entiers, chaînes de caractères, que des tableaux ou tables de hachage.

la session n'est valable que durant un laps de temps limité (en général 30 minutes, ce qui peut être modifié dans le fichier de configuration d'Apache), et elle est automatiquement détruite à la fermeture du navigateur du client.

2.8.1 Démarrer une session

Il existe trois moyens de démarrer une session :

- *Démarrage automatique* : Dans le fichier `php.ini`, en passant le paramètre `session.auto_start` à `1`, une session démarrera automatiquement à chaque nouvelle connexion d'un visiteur sur le site.
- *Démarrage explicite* : La fonction `session_start()` permet, comme son nom l'indique, de démarrer une session. Si le visiteur appelle la page pour la première fois, un identifiant lui est attribué, sinon, son identifiant est récupéré. Cette fonction devra être présente sur toutes les pages du site où l'on souhaite utiliser les variables du visiteur.
- *Démarrage implicite* : La fonction `session_register()` permet de se passer du démarrage de la session. Elle stocke des variables dans la session ; si la session n'est pas ouverte, elle en ouvre une. Par exemple, pour stocker les variables `$a` et `$b` dans la session :
  `session_register("a", "b");`

2.8.2 Accéder aux variables d'une session

Les variables de session sont stockées dans la table de hachage `$_SESSION`. Ainsi, pour accéder à la variable `$a` de l'exemple précédent, nous appelerons `$_SESSION{"a"}`. Voici un exemple très simple où l'on stocke des variables dans la session et on les réaffiche :

```
<?PHP
  session_start();
  $var1="Variable de session PHP";
  $var2=123;
  session_register("var1", "var2");
  print "Les variables de sessions :<BR />\n";
  print "  var1 : ".$_SESSION{"var1"}."<BR />\n";
  print "  var2 : ".$_SESSION{"var2"}."<BR />\n";
?>
```

2.8.3 Quelques fonctions sur les sessions

Voici présentées brièvement les fonctions permettant de manipuler une session.

2.8.3.1 session_destroy()

Détruit la session en cours en conservant les variables. Pour fermer proprement une session, il faut l'associer à `session_unset()`.

2.8.3.2 *boolean* session_is_registered(*string* NomVariable)

Teste si une variable fait partie de la session.

```
if (session_is_registered("var1"))
  print "var1 est définie.<BR />\n";
else
  print "var1 indéfinie dans la session.<BR />\n";
```

2.8.3.3 *string* session_save_path([*string* Chemin])

Renvoie le chemin de sauvegarde des sessions. Si *Chemin* est spécifié, modifie le chemin de sauvegarde des sessions.

2.8.3.4 session_unregister(*string* NomVariable)

Détruit la variable *NomVariable* de la session active.

```
session_unregister("var1");
```

2.8.3.5 session_unset()

Détruit toutes les variables de la session active.

2.9 Notions de Programmation Orientée-Objet

Cette partie n'est en aucun cas une introduction aux méthodes de Programmation Orientée-Objet : je considèrerai que la Philosophie "Objet" est déjà acquise.
En PHP, la définition d'une classe se fait par `class` et la définition de ses attributs par `var`[8]. A l'intérieur de la classe ces attributs sont accessibles via la variable `$this` qui désigne l'objet courant. Ne pas oublier de créer une méthode (fonction), dite "constructeur", qui porte le même nom que la classe et permet de créer un objet de cette classe. Exemple :

```
class Polygone
{
 var $pi=3.14;
 var $nb_pts;
```

[8]Il est possible de définir des attributs statiques en les initialisant lors de la définition.

```
 function Polygone($nb_pts)
 {
   $this->nb_pts=$nb_pts;
 }
}
```

La création d'un nouvel objet Polygone se fait par :

```
$losange=new Polygone(4);
```

On pourra accéder aux variables de l'objet par le symbole `->`.

```
print $losange->nb_pts."<BR />\n";
```

Pour détruire cet objet on utilise la fonction `unsert()` :

```
unsert($losange);
```

La notion d'héritage est obtenue par le mot-clé `extends`.

```
class Triangle extends Polygone
{
 function Triangle($nb_pts)
 {
   if ($nb_pts != 3)
   {
     print "Triangle = 3 sommets ... et non $nb_pts !<BR />\n";
     exit;
   }
   $this->Polygone(3);
 }
}
```

Toutes les méthodes de Polygone sont accessibles pour Triangle mais peuvent être redéfinies (définition d'une fonction de même nom dans la classe Triangle).

On peut également définir des classes génériques définissant des méthodes statiques (un appel à une méthode de cette classe ne nécessite pas la création d'un objet). Les appels se font alors par `nom_de_classe::nom_de_méthode([arguments])`.

```
class calculs
{
 var $pi=3.14;

 function diam_cercle($rayon)
 {
   return(2*$pi*$rayon);
 }
}

$r=15;
$diametre=calculs::diam_cercle($r);
```

Le modèle objet présenté précédemment est celui de PHP4. Voici les nouveautés de PHP5[9].
Les fonctions de construction et de destruction portent maintenant des noms spéciaux : `__construct()` et `__destruct()`. Dans les cas d'héritage, l'appel au constructeur parent se fait par `parent::__construct()`.
Les notions de visibilité pour les objets et les méthodes ont été ajoutées :

- *public* : on peut accéder à l'attribut ou la méthode de puis n'importe où.
- *protected* : on ne peut accéder à l'attribut ou à la méthode que par des classes descendantes.
- *private* : on ne peut accéder à l'attribut ou à la méthode que dans la classe de définition.

Vous pourrez trouver plus de précisions sur internet ou dans Linux Magazine Hors série n°20.

2.10 Récupérer les informations d'un formulaire

Il existe deux méthodes permettant de transmettre des informations via un formulaire (balise `<FORM>`) :

- La méthode GET : les différents champs sont transmis dans l'URL après un caractère "`?`". Ils sont sous la forme *Nom_Champs=Val* et séparés par le caractère "`&`". Les caractères d'espacement sont remplacés par des "`+`". Prenons l'exemple du formulaire suivant (annuaire téléphonique) :

```
<FORM action="exec.php" method=GET>
 Nom : <INPUT type=text size=15 name=Nom /><BR />
 Prénom : <INPUT type=text size=15 name=Prenom /><BR />
 <INPUT type=submit value="Ok !" />
</FORM>
```

  Si nous saisissons "Geller" dans le champs Nom et "Ross" dans le champs Prénom, l'URL appelée sera :
  `http://.../exec.php?Nom=Geller&Prenom=Ross`
- La méthode POST transmet les informations de manière tranparente pour l'utilisateur et on évite ainsi de générer des URL très longues.

En général la méthode POST est préférée à la méthode GET. Toutefois, il peut être intéressant d'utiliser des méthodes GET pour permettre à l'utilisateur de générer automatiquement les adresses de requêtes qui l'intéresse et permettre d'effectuer du web mining (fouille de données sur le web).
Les informations sont transmises dans une table de hachage : `$_GET` pour la méthode GET et `$\_POST` pour la méthode POST. D'après l'exemple précédent, si nous voulons récupérer la valeur du champs Nom dans le script `exec.php`, nous exécuterons :

```
$nom=$_GET['Nom'];
```

Là où les choses se compliquent, c'est lorsque vous désirez transmettre directement une structure plus complexe comme par exemple un tableau. Il faudra alors "sérialiser" l'information par `serialize()`, c'est-à-dire qui permet d'obtenir une représentation linéaire des informations. Pour pouvoir être utilisée lors de la récupération il faudra alors "désérialiser" l'information par `unserialize()`. Considérons le tableau suivant :

[9] Mais vous pouvez continuer à utiliser le modèle PHP4 par raison de compatibilité ascendante.

```
$tab{"Rachel"}="Green";
$tab{"Monica"}="Geller";
$tab{"Phoebe"}="Buffay"
```

La sérialisation de ce tableau (qui sera stockée dans une variable), indiquant la structure et les données contenues dans le tableau est :

```
a:3:{s:6:"Rachel";s:5:"Green";s:6:"Monica";s:6:"Geller";s:6:"Phoebe";
s:6:"Buffay";}
```

`a:3` indique un tableau de trois éléments et les `s:x` indiquent des chaînes de caractères (*string*) de x caractères. En stockant ce tableau serialisé dans la variable `$serial_tab` par `$serial_tab=serialize($tab)`, nous pourrons transmettre ces informations dans un formulaire grâce au type hidden :

```
<INPUT type=hidden value=$serial_tab name=table />
```

Par la suite, dans le script réceptionnant les informations du formulaire, les données pourront être récupérées par :

```
$table=unserialize(stripSlashes($_POST['table']));
```

La fonction stripSlashes() est importante car, si le tableau contient des chaînes de caractères possédant des caractères tels que (”), (’), ou (\), alors, lors de la transmission des données ces caractères seront protégés par un symbole (\) supplémentaire et il faudra les supprimer par la suite.

2.11 Gestion des Erreurs

La fonction `error_reporting()` permet de gérer le type d’affichage des erreurs. Les différents types d’erreur sont définis par des constantes dont voici les principales :

Constante	Description
E_ERROR	Erreur irrécupérable
E_WARNING	Erreur récupérable
E_NOTICE	Erreur possible
E_PARSE	Erreur du parser
E_ALL	Toutes les erreurs

Ces différents types d’erreurs peuvent être liés à l’aide de connecteurs logiques ET (caractère `&`), OU (caractère `|`), et NON (caractère `~`).

2.12 A propos des Warnings

Les Warnings sont très importants puisqu’ils vont nous permettre de débugger le code. Toutefois, dans certains cas, certaines fonctions peuvent renvoyer des Warnings que l’on ne désire pas afficher (par exemple, pour tester si un utilisateur est enregistré dans une base de donnée MySQL). Pour désactiver le Warning provoqué par l’appel de la-dite fonction, il faut alors la faire précéder du signe ”`@`”. Exemple :

```
@print "Pas de Warning provoqué par l'appel de print";
```

2.13 Recommandations de programmation

N'oubliez pas d'indenter correctement votre code et le code HTML produit.
N'hésitez pas à user et abuser du découpage de votre code en modules grâce à la fonction `require_once()` qui vous permet d'insérer dans votre programme du code PHP issu d'un autre fichier. Ou alors, vous pouvez également utiliser la Programmation Orientée-Objet. Séparez alors chaque classe dans un fichier séparé que vous incluerez encore à l'aide de `require_once()`.
L'insertion de commentaires dans votre code (surtout au niveau des définitions de fonctions) permet de réutiliser/corriger des anciens codes beaucoup plus rapidement.

3

PHP & MySQL

I am Linus Torvalds, and yes I am your God.
Linus Torvalds

Le Système de Gestion de Bases de Données le plus utilisé avec PHP est MySQL. Nous allons tout d'abord étudier ce système de manière isolée puis en liaison avec PHP.

3.1 MySQL

MySQL peut être utilisé de deux façons : en mode console ou en mode graphique via l'interface PHPMyAdmin. Le mode graphique est bien sûr plus simple d'utilisation, notamment pour la construction de requêtes SQL mais lorsque nous travaillerons avec PHP nous serons en mode console. Il est donc intéressant de maîtriser ces deux modes d'accès à MySQL. Notez toutefois que le but de cette partie n'est pas de donner un cours de SQL !

3.1.1 Le mode console

La première étape est bien sûr la connexion à la base. Elle s'effectue à l'aide d'un login et d'un mot de passe. Si vous n'avez pas modifier la configuration par défaut de MySQL, l'utilisateur `root` n'a pas de mot de passe. La syntaxe de connexion est :

```
mysql -u login_utilisateur -p [nom_base]
```

Pour créer une nouvelle base de travail, nous utiliserons la commande `CREATE DATABASE [Nom]` et pour modifier les droits sur cette base : `GRANT`. Voici par exemple comment créer une nouvelle base et un nouvel utilisateur :

```
CREATE DATABASE Base_Test;
GRANT ALL PRIVILIEGES ON Base_Test.* TO Administrateur_Test@localhost
 IDENTIFIED BY 'Mot_de_passe';
```

On a créé une base *Base_Test* et un utilisateur *Administrateur_Test* ayant pour mot de passe *Mot_de_passe*. Pour se connecter à cette base, l'utilisateur pourra alors rentrer :

```
mysql -u Administrateur_Test -p Base_Test
```

Il est possible de créer un fichier de configuration `.my.cnf` permettant d'éviter d'avoir à taper les mots de passe. La structure de ce fichier est :

```
[client]
user=nom_utilisateur
password=mot_de_passe
```

La lecture d'un fichier contenant des requêtes SQL se fait par une redirection ou par la commande `source` :

```
mysql < fichier.sql
```

ou `source fichier.sql` dans MySQL

Les autres commandes sont les fonctions SQL classiques. Rappel : les tables sont indexées sur la clé primaire ; le choix judicieux de cette clé et de son type (entier par exemple) permet d'améliorer les temps d'acès à la base lors des requêtes.

3.1.2 PHPMyAdmin

PHPMyAdmin[10] est un outil entièrement écrit en PHP et qui permet d'administrer de manière simple une base MySQL. La plupart des commandes MySQL sont accessibles par l'intermédiaire de MySQL. Ce que l'on peut faire avec PHPMyAdmin :

- Créer et détruire des bases de données (en root).
- Créer, Modifier et Détruire des schémas de tables.
- Consulter le contenu des tables.
- Modifier/Détruire le contenu des lignes des tables.
- Exécuter des requêtes SQL de manière interactive (très utile pour tester une requête avant de l'intégrer dans du code PHP).
- Charger des données dans les tables depuis des fichiers.
- Gérer les utilisateurs MySQL.

Un exemple de session PHPMyAdmin (création d'une requête de manière interactive) est donné en figure 3.5

[10] Son installation se fait tout simplement par apt-get install phpmyadmin.

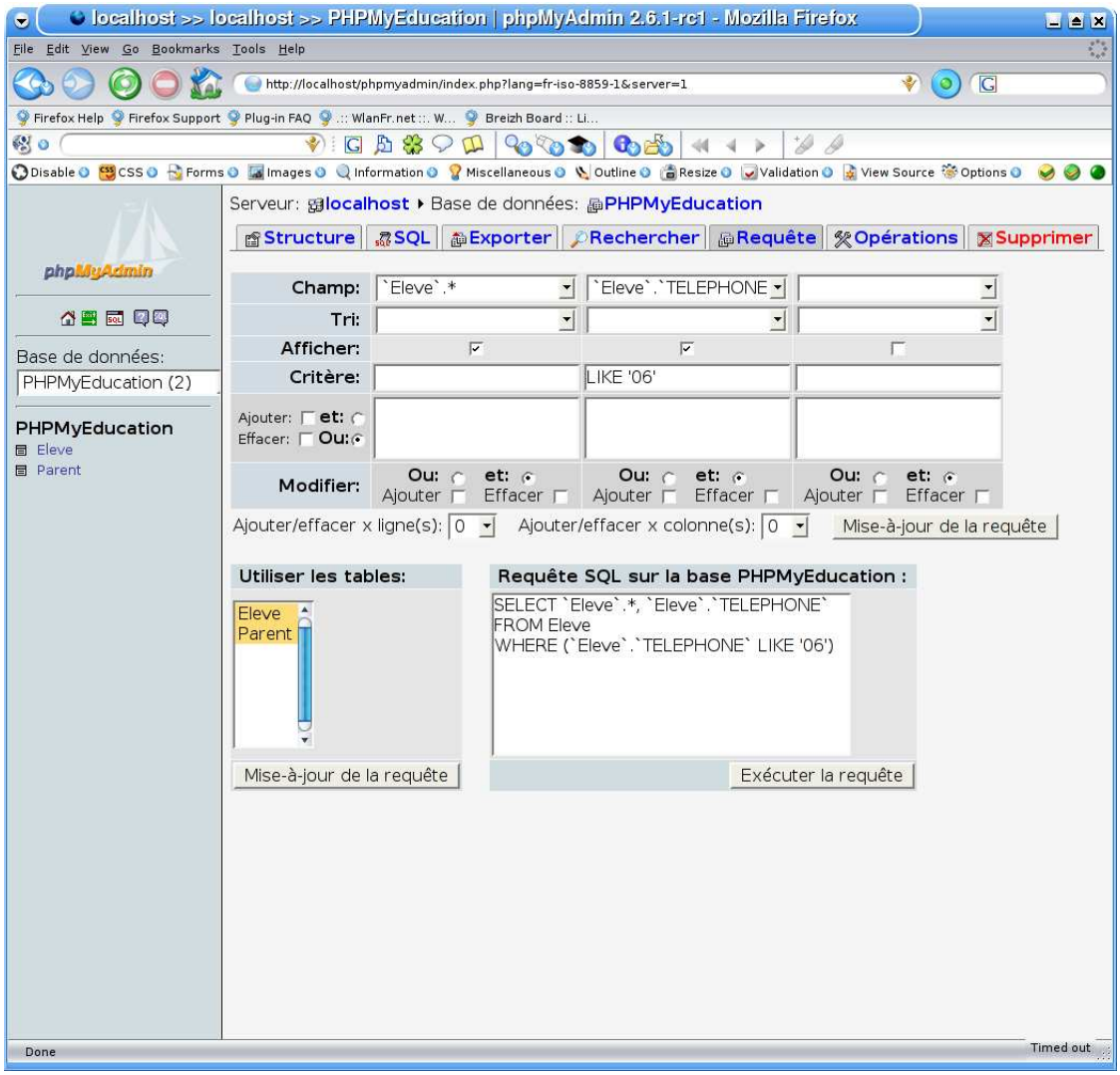


FIG. 3.5 – Interface PHPMyAdmin

3.2 PHP et MySQL

Abordons maintenant la liaison de PHP avec MySQL. Les opérations fondamentales en PHP sont assez simples : c'est surtout la requête SQL qui sera importante ! Les opérations PHP sont au nombre de deux : la connexion à la base (la déconnexion est automatique) et la requête dans la base. Il est très intéressant ici d'effectuer une programmation modulaire permettant de récupérer les fonctions qui seront utiles à chaque fois que l'on désirera accéder à une base.

3.2.1 Connexion

Pour se connecter à une base de données, il faut fournir quatre paramètres : le nom de l'utilisateur MySQL, son mot de passe, le nom de la base et le nom du serveur (le serveur mysqld n'est pas forcément sur la même machine que le serveur Apache – cf Chapitre 1).
Connexion.php :

```
<!--
  Module de Connexion à une base MySQL
-->

<?PHP
  function Connexion($Nom, $MotPasse, $Base, $Serveur)
  {
    // Connexion au serveur MySQL
    $connexion=mysql_pconnect($Serveur, $Nom, $MotPasse);
    // Gestion des erreurs
    if (empty($connexion))
    {
      print "Echec de connexion au serveur $Serveur !<BR />\n";
      exit;
    }
    // Connexion à la base et gestion des erreurs
    if (!mysql_select_db($Base, $connexion)
    {
      print "Accès à la base $Base impossible : ".mysql_error($connexion)."<BR />\n";
      exit;
    }
    // Renvoi de la variable de connexion
    return($connexion)
  }
?>
```

La variable `$connexion` renvoyée par la fonction est la variable identifiant la connexion à la base de données : c'est elle qui permettra d'effectuer des requêtes. Pour se connecter à la base locale

(*localhost*) *MiddleGard* en tant que *Gollum* (mot de passe : 'Precieuuux'), on effectuera :

```
require_once("Connexion.php");
$connexion=Connexion("Gollum", "Precieuuux", "MiddleGard", "localhost");
```

Mais dans un projet important en PHP, il faudra ouvrir la (ou les) base(s) dans plusieurs scripts ... et on risque de mélanger les paramètres de connexion. Pour résoudre ce problème, on peut créer un fichier dans lequel on définira des constantes qui nous serviront lors de la connexion. En utilisant les paramètres précédents, on créera par exemple :
Param_Connexion_MiddleGard.php :

```
<!--
  Paramètres de connexion à MiddleGard
-->

<?PHP
  define("NOM", "Gollum");
  define("PASSE", "Precieuuux");
  define("BASE", "MiddleGard");
  define("SERVEUR", "localhost");
?>
```

La connexion se fera alors simplement par :

```
require_once("Connexion.php");
require_once("Param_Connexion_MiddleGard.php");
$connexion=Connexion(NOM, PASSE, BASE, SERVEUR);
```

3.2.2 Requête

L'exécution d'une requête SQL produit le retour d'un objet MySQL contenant le résultat sous forme d'une table. Il faut donc être capable d'exécuter une requête et de lire le résultat :
Requete.php :

```
<!--
  Exécution de requêtes
-->

<?PHP
  function Requete($requete, $connexion)
  {
    // Exécution de la requête
    $resultat=mysql_query($requete, $connexion);
    // Gestion des erreurs
    if ($resultat)
      return($resultat);
```

```
    else
    {
      print "Erreur dans l'exécution de la requête : $requete<BR />\n";
      print "Erreur MySQL : ".mysql_error($connexion)."<BR />\n";
      exit;
    }
  }

  function EltSuivant($resultat)
  {
    // Renvoi l'elt suivant du résultat de la requête
    return(mysql_fetch_object($resultat));
  }

  function LigneSuivante($resultat)
  {
    // Renvoi la ligne suivante du résultat de la requête
    return(mysql_fetch_assoc($resultat));
  }
?>
```

3.3 Différentes façons de lire un résultat de requête

Après une requête du type : `$resultat=mysql_query($requete, $connexion);`, il y a plusieurs façons de lire le résultat. Dans cette partie, nous considèrerons que la requête renvoie 3 lignes contenant les champs NOM, PRENOM et ADRESSE.
Tout d'abord, comme nous l'avons vu précédemment, il y a la fonction `mysql_fetch_object()` qui renvoie un objet contenant les champs de la requête. Ici, avec un appel du type `$elt=mysql_fetch_object()` nous aurions les objets :
`$elt->NOM`, `$elt->PRENOM` et `$elt->ADRESSE`.
Ensuite, `mysql_fetch_row()` récupère une ligne de la requête sous forme de tableau. Avec `$elt=mysql_fetch_row()`, `$elt[0]` contient NOM, `$elt[1]` contient PRENOM, ...
`mysql_fetch_array()` renvoie une table de hachage indicée par le nom des colonnes de la requête. `$elt=mysql_fetch_array()` renvoie `$elt{NOM}`, `$elt{PRENOM}`, ...

3.4 PHP et ORACLE

C'est bien contraint et forcé que je décrirai rapidement quelques commandes relatives à la base de donnée propriétaire ORACLE. Comme précédemment, nous ne verrons que les commandes vraiment essentielles.

3.4.1 Connexion

Deux méthodes de connexion sont ici disponibles : une méthode non persistante (fermeture automatique de la base à la fin du script PHP)
`$connexion=OCILogon(nom_utilisateur, mot_de_passe, nom_base)` et une méthode persistante `OCIPLogon()` qui se comporte comme la précédente.

3.4.2 Requête

Par défaut, PHP fonctionne en mode `COMMIT_ON_SUCCESS` (auto-commit-mode). Il faut appliquer deux fonctions : l'une pour créer la requête et l'autre pour l'exécuter :

```
$connexion=OCILogon("Moi", "MonMotDePaSse", "MaBase");
$requete=OCIParse($connexion, "SELECT ....");
OCIExecute($requete);
```

Tant que le commit n'est pas effectué (si le commit automatique a été désactivé), l'état de la base reste inchangé (en cas de modification bien sûr). Il faut faire :

```
OCICommit($connexion);
```

Pour lire les requêtes, on effectue comme avec MySQL une boucle sur les résultats :

```
while (OCIFetch($requete))
  print OCIResult($requete, "NOM_COLONNE")."\n";
```

Voici maintenant un exemple plus complet faisant intervenir la fonction `OCIError()` d'affichage des erreurs.

```
<?PHP
  $connexion=OCILogon("Moi", "MoTdEpAsSe", "BASE");
  if (empty($connexion))
  {
    $erreur=OCIError();
    print "Erreur ".$erreur{"code"}.": ".$erreur{"message"}."\n";
    exit;
  }
  $requete=OCIParse($connexion, "... requete ...");
  if (empty($requete))
  {
    $erreur=OCIError();
    print "Erreur ".$erreur{"code"}.": ".$erreur{"message"}."\n";
    exit;
  }
  if (!OCIExecute($requete))
  {
    $erreur=OCIError();
    print "Erreur ".$erreur{"code"}.": ".$erreur{"message"}."\n";
    exit;
```

```
  }
  while (OCIFetch($requete))
    print OCIResult($requete, "NOM_COLONNE")."\n";
?>
```

Et voici en deuxième exemple, une fonction permettant d'afficher un tableau HTML d'une table `$table` de la base `$base`.

```
function oracle_HTML($base , $table)
{
  $requete = "SELECT * FROM $table";
  $types = array("NUMBER", "DATE");
  if ($res = OCIParse($base, $requete))
  {
    OCIExecute($res);
    if (OCIError($res))
      return OCIError($res);
    else
    {
      $ncols = OCINumCols($res);
      $retour = array();
      $output = "<table>\n\t<tr>\n";
      for ($i = 1; $i <= $ncols; $i++ )
        $output .= "\t\t<td align=center>".OCIColumnName($res, $i)."</td>\n";
      while(OCIFetch($res))
      {
        $output .= "<tr>\n";
        for ($i = 1; $i <= $ncols; $i++ )
        {
          if (in_array(OCIColumnType($res, $i), $types))
            $output .= "\t\t<td align=center>".OCIResult($res, $i)."</td>\n";
          else
            $output .= "\t\t<td>".OCIResult($res, $i)."</td>\n";
        }
        $output .= "\t</tr>\n";
      }
      $output .= "</table>";
    }
  }
  else
    return OCIError($res);
  return $output;
}
```

On notera l'utilisation de quelques nouvelles fonctions :

- `OCINumCols(Résultat)` retourne le nombre de colonnes de *Résultat*.
- `OCIColumnType(Résultat, Numéro)` retourne le type de données de la colonne correspondant au *Numéro* de colonne dans le *Résultat* (les colonnes sont indexées à partir de 1).
- `in_array(Valeur, Tableau)` renvoie Vrai si la *Valeur* appartient au *Tableau*. Donc, dans l'exemple précédent, si l'une des colonnes du résultat est de type `NUMBER` ou `DATE`, on affichera le résultat de manière centrée.

3.5 Conclusion

Les commandes d'accès à une base MySQL sont très simples, de même que l'exécution et la récupération du résultat d'une requête. Les plus grosses sources d'erreurs se situent donc au niveau de la base : requêtes SQL mal formées, ou pire, mauvais schéma de base. Portez donc une très grande attention à l'élaboration de votre schéma de base ! Je vous conseille d'ailleurs de toujours créer un diagramme UML de votre base (grâce au logiciel Dia notamment) : ceci pour éviter de perdre du temps en phase de développement (création des requêtes SQL) et pour pouvoir reprendre votre code (ou permettre à d'autres développeurs de le reprendre) après l'avoir abandonné quelques temps ...

4

Librairies PHP

Computers are like Old Testament gods ; lots of rules and no mercy.
Joseph Campbell

DANS ce chapitre je présenterai deux librairies très utilisées en PHP : la librairie FPDF qui permet de générer à la volée des documents PDF et la librairie graphique JPGraph, surcouche de la librairie graphique native du PHP, permettant de générer sans trop de difficulté des graphes relativement complexes.

4.1 La librairie FPDF

La FPDF est une librairie Orientée-Objet libre (le F de FPDF signifie Free !) permettant de générer des documents PDF et disponible sur `http://www.fpdf.org`. Cette librairie se compose d'un fichier de classe `fpdf.php` (qu'il faudra inclure à l'aide d'une commande `require` lorsque vous désirerez l'utiliser) et des fichiers de définition des polices. Il faudra indiquer où se trouve le répertoire contenant les polices dans la constante `FPDF_FONTPATH`. Donc, typiquement, les premières lignes d'un script PHP utilisant la librairie FPDF seront :

```
define(FPDF_FONTPATH, "/home/data/font/");
require("fpdf.php");
```

La première étape consiste a créer un objet FPDF. Cet objet dépend de trois paramètres :

- *l'orientation de la page*, qui peut être en portrait (valeur 'P') ou en paysage ('L'),
- *l'unité de mesure*, qui peut être mm, cm, in, pt,
- *le format de la page*, parmi A3, A4, A5, ...

Voici un exemple de construction d'un objet FPDF : `$pdf=new FPDF('P', 'mm', 'A4');` Les valeurs choisies ici sont d'ailleurs les valeurs par défaut, ce qui signifie que `$pdf=new FPDF();` aura le même effet.

Il faut ensuite ajouter une page (pour l'instant nous n'avons créé qu'un objet vide). Cela se fait

par la méthode `AddPage()`.
`$pdf->AddPage();`
Par défaut, l'origine du texte est en haut à gauche à 1 cm des bords.
Avant de pouvoir imprimer un texte, il est impératif de définir la police qui sera utilisée sinon le document serait invalide. Ce choix s'effectue par la méthode `SetFont()` qui dépend de trois paramètres :

- *le type de police* parmi Arial, Times, Courier, Symbol et ZapfDingBats,
- *le style de la police* qui sera une combinaison de gras ('B'), italique ('I'), souligné ('U') ou normal (''),
- *la taille* donnée en points.

Pour prendre une police de type Arial, 16pts, en gras, il faut taper :
`$pdf->SetFont('Arial', 'B', 16);`
L'impression se fait ensuite par un système de cellules avec la méthode `Cell()`. Une cellule est une zone rectangulaire, éventuellement encadrée, qui contient du texte. Elle est imprimée à la position courante. Pour simplifier, la méthode `Cell()` accepte six arguments :

- *La longueur* de la cellule exprimée dans l'unitée choisie lors de la création de l'objet FPDF,
- *la largeur* de la cellule exprimée dans l'unitée choisie,
- *le texte*,
- *l'encadrement* : 1 pour encadrer la cellule et 0 sinon (paramètre optionnel).
- *le retour à la ligne* : 1 pour retourner à la ligne et 0 sinon (paramètre optionnel). Le retour à la ligne peut être effectué par la méthode `Ln()`.
- *le positionnement* : 'C' pour centré, 'L' pour alignement gauche et 'R' pour alignement droit (paramètre optionnel).

Voici un texte encadré et centré :
`$pdf->Cell(150, 10, 'Voici mon Texte !', 1, 0, 'C');`
Lorsqu'une cellule descend trop bas (2 cm du bas de page par défaut), un saut de page automatique est effectué. Pour régler le seuil de déclenchement du saut de page, on utilise la méthode `SetAutoPageBreak()`.

Et enfin, pour produire un affichage, il faut utiliser la méthode `Output()`. Sans paramètre l'affichage se fera à l'écran (veillez alors à ce qu'il n'y ait aucun caractère, pas même un espace, avant votre balise `<?PHP` et après votre balise `?>` : il y aurait un message d'erreur ou une page blanche !). En indiquant un nom de fichier, le document pourra être sauvegardé dans ce fichier.

Pour définir un document plus complexe, on créera un nouvel objet héritant de FPDF (`class PDF extends FPDF` par exemple) et dans lequel on pourra définir par exemple les méthodes `Header()` et `Footer()` permettant de définir un en-tête et un pied de page.

4.1.1 Quelques méthodes de FPDF

Je présente ici brièvement quelques méthodes utiles de FPDF. On supposera l'existence d'un objet FPDF nommé `$pdf`. Des exemples d'utilisation de ces méthodes seront données dans la partie "Exemple complet".

4.1.1.1 *int* $pdf->Addlink()

Crée un nouveau lien interne et renvoie son identifiant. Un lien interne est une zone cliquable qui amène à un autre endroit dans le document. La destination est définie à l'aide de `SetLink()`.

4.1.1.2 $pdf->AliasNbPages([*string* alias])

Définit un alias indiquant le nombre total de pages du document. Par défaut, cet alias se nomme `{nb}`.

4.1.1.3 $pdf->Image(*string* Nom, *float* x, *float* y [, *float* w [, *float* h [, *string* type [, *mixed* link]]]])

Affiche une image au format PNG ou JPEG. Les paramètres sont :
- *Nom* : le nom du fichier image,
- *x, y* : les coordonnées d'affichage du coin supérieur gauche,
- *w* : largeur de l'image dans la page,
- *h* : hauteur de l'image dans la page,
- *type* : format de l'image (JPEG, JPG, ou PNG),
- *link* : URL (lien externe) ou identifiant renvoyé par `Addlink()` (lien interne vers une page) et vers lequel pointe l'image.

4.1.1.4 $pdf->MultiCell(*float* w, *float* h, *string* txt [, *mixed* border [, *string* align [, *int* fill]]])

Cette méthode permet d'imprimer du texte avec des retours à la ligne. Ceux-ci peuvent être automatiques (dès que le texte atteint le bord droit de la cellule) ou explicites (via le caractère `\n`). Le texte peut être aligné, centré ou justifié. Le bloc de cellules peut être encadré et le fond coloré. Les paramètres sont :
- *w* : la largeur des cellules. Si elle vaut 0, elles s'étendent jusqu'à la marge droite de la page.
- *h* : la hauteur des cellules.
- *txt* : la chaîne à imprimer.
- *border* : indique si des bords doivent être tracés autour du bloc de cellules (0 : aucun bord, 1 : cadre).
- *align* : Contrôle l'alignement du texte. Les valeurs possibles sont : 'L' pour alignement à gauche, 'C' pour centrage, 'R' pour alignement à droite, et 'J' pour justification (valeur par défaut).
- *fill* : Indique si le fond des cellules doit être coloré '1' ou transparent '0'. La valeur par défaut est '0'.

4.1.1.5 $pdf->PageNo()

Renvoie le numéro de la page courante.

4.1.1.6 $pdf->SetDrawColor(*int* r [, *int* g, *int* b])

Fixe la couleur pour toutes les opérations de tracé (lignes, rectangles et contours de cellules). Elle peut être indiquée en composantes RGB (de 0 à 255) ou en niveau de gris (si seulement le premier paramètre est indiqué).

4.1.1.7 $pdf->SetFillColor(*int* r [, *int* g, *int* b])

Fixe la couleur pour toutes les opérations de remplissage (rectangles pleins et fonds de cellules). Elle peut être indiquée en composantes RGB (de 0 à 255) ou en niveau de gris (si seulement le premier paramètre est indiqué).

4.1.1.8 $pdf->SetTextColor(*int* r [, *int* g, *int* b])

Fixe la couleur pour le texte. Elle peut être indiquée en composantes RGB (de 0 à 255) ou en niveau de gris (si seulement le premier paramètre est indiqué).

4.1.1.9 $pdf->SetX(*float* x)

Fixe l'abscisse de la position courante. Si la valeur transmise est négative, elle est relative à l'extrémité droite de la page.

4.1.1.10 $pdf->SetY(*float* y)

Ramène l'abscisse courante à la marge gauche et fixe l'ordonnée. Si la valeur transmise est négative, elle est relative au bas de la page.

4.1.2 Exemple complet

```
<?php
  define(FPDF_FONTPATH, "/usr/share/fpdf/font/");
  require("/usr/share/fpdf/fpdf.php");

  class PDF extends FPDF
  {

   var $titre;
```

```
function PDF($orientation="P", $mesure="mm", $page="A4", $titre="Titre")
{
 $this->FPDF($orientation, $mesure, $page);
 $this->titre=$titre;
 // L'alias {nb} contiendra le nombre total de pages
 $this->AliasNbPages();
}

function Header()
{
 // Police Arial Gras 16 pts
 $this->SetFont('Arial', 'B', 16);
 // Cadre en rouge
 $this->SetDrawColor(255, 0, 0);
 // Font en noir
 $this->SetFillColor(0, 0, 0);
 // Texte en bleu
 $this->SetTextColor(0,0,255);
 // Calcul de la largeur du titre pour centrer la cellule
 $w=$this->GetStringWidth($titre)+100;
 $this->SetX((210-$w)/2);
 // Affichage du titre au centre du cadre
 $this->Cell($w, 9, $this->titre, 1, 1, 'C', 1);
 // Saut de ligne de 10 mm
 $this->Ln(10);
}

function Footer()
{
 // On se positionne à 15 mm du bas de page
 $this->SetY(-15);
 // Police Arial italique 8 pts
 $this->SetFont("Arial", "I", 8);
 // Couleur gris moyen pour le texte
 $this->SetTextColor(128);
 // Affichage du numéro de page
 $this->Cell(0, 10, "Page n°".$this->PageNo()."/{nb}", 0, 0, "C");
}

function AfficheTexte($fichier)
{
 // Gestion des erreurs
 if (empty($fichier))
 {
```

```
     print "Vous devez spécifier un nom de fichier !<BR />\n";
     exit;
    }
    // Lecture du fichier
    $file=fopen($fichier, "r");
    if (empty($file))
    {
     print "Impossible d'ouvrir $fichier ...<BR />\n";
     exit;
    }
    $txt=fread($file, filesize($fichier));
    fclose($file);
    // Choix de la police Times 12 pts
    $this->SetFont("Times", "", 12);
    // Afficahge du texte
    $this->MultiCell(0, 5, $txt);
    // Retour en marge gauche
    $this->SetX(0);
   }

  }

  $pdf=new PDF("P", "mm", "A4", "Exemple Complet");
  $pdf->AddPage();
  $pdf->AfficheTexte("Exemple.txt");
  $pdf->Output();
 ?>
```

4.2 La librairie graphique JPGraph

La librairie JPGraph est une librairie graphique Orientée-Objet libre, disponible par `apt-get install libphp-jpgraph` (notez aussi la possibilité de rapatrier des exemples d'application par `apt-get install libphp-jpgraph-examples`). Elle permet de créer des graphiques complexes avec un minimum de code. Pour celà elle s'appuie sur le module graphique de PHP, le module GD qui doit être également installée (`apt-get install php4-gd` ou `apt-get install php5-gd` suivant la version de PHP que vous utilisez). Les principales caractéristiques de la librairie JPGraph sont :

- Le support des formats GIF, JPEG, PNG, et TTF.
- La mise à disposition de différents types de graphiques : digrammes en lignes, en barres, camemberts, ...
- La possibilité d'utiliser un module de cache pour minimiser l'utilisation du temps processeur.

Les images générées par la JPGraph seront incluses dans les documents HTML dans une balise `<IMG ... />`. Tout comme pour les fichiers PDF générés à la volée, les fichiers images ne devront comportés que des lignes de code PHP générant l'image. Par exemple, si c'est le script image1.php qui génère l'image, l'appel se fera par :

```
<IMG src="image1.php" />
```

Astuce : on pourra même passer des paramètres au script en utilisant une méthode GET (valeur dans l'adresse) :

```
<IMG src="image1.php?arg1=...&argn=..." />
```

4.2.1 Générer une image

Pour générer une image il faudra include la librairie JPGraph et au moins l'une des autres librairie de sortie graphique parmi : jpgraph_bar (pour des diagrammes à barres), jpgraph_line, jpgraph_pie, jpgraph_pie3d, jpgraph_scatter, et jpgraph_spider, jpgraph_gantt.
S'agissant d'une librairie Orientée-Objet, il faut dans un premier temps créer un objet `Graph`. Ensuite, on calculera le graphique et on l'affichera avec la méthode `Stroke()`. Je vais maintenant présenté des exemples de diagrammes et le résultat obtenu. Dans tous ces exemples, les données utilisées seront stockées dans un tableau `$data`.
Pour pouvoir afficher ces exemples il faudra deux fichiers : le fichiers du script PHP générant le graphe proprement dit, et le fichier HTML ou PHP contenant la balise `<IMG ...\>`.

4.2.1.1 Lignes

```
<?PHP
  include("/usr/share/jpgraph/jpgraph.php");
  include("/usr/share/jpgraph/jpgraph_line.php");

  $data=array(12, 5, 8, 25, 1, 14);

  // Nouvel objet Graph
  $graph=new Graph(300, 200);
  $graph->SetScale("textlin");
  // Nouvelle ligne composée des points de $data
  $line1=new LinePlot($data);
  // Ajout de la ligne $line1 dans le graphe
  $graph->Add($line1);
  // Affichage du graphe
  $graph->Stroke();
?>
```

On a créé un objet `Graph` de longueur 300 et de hauteur 200, puis une ligne composée des points du tableau `$data`. On a ajouté cette ligne au graphe et on a affiché le tout. Le résultat est visible en figure 4.6.

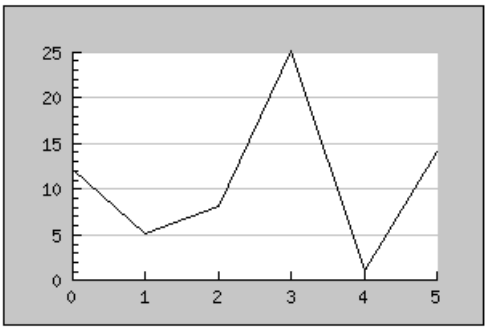


FIG. 4.6 – Diagramme de lignes

4.2.1.2 Barres

```
<?PHP
  include("/usr/share/jpgraph/jpgraph.php");
  include("/usr/share/jpgraph/jpgraph_bar.php");

  $data=array(12, 5, 8, 25, 1, 14);

  // Nouvel objet Graph
  $graph=new Graph(300, 200);
  $graph->SetScale("textlin");
  // Nouvelle barre composée des points de $data
  $bar1=new BarPlot($data);
  // Ajout de la barre $bar1 dans le graphe
  $graph->Add($bar1);
  // Affichage du graphe
  $graph->Stroke();
?>
```

La seule différence par rapport à l'exemple précédent est que l'on va générer des barres avec `BarPlot()` (figure 4.7). On aurait pu mixer les deux exemples pour obtenir la figure 4.8.

4.2.1.3 Camemberts

```
<?PHP
  include ("/usr/share/jpgraph/jpgraph.php");
  include ("/usr/share/jpgraph/jpgraph_pie.php");

  $data=array(12, 5, 8, 25, 1, 14);
```

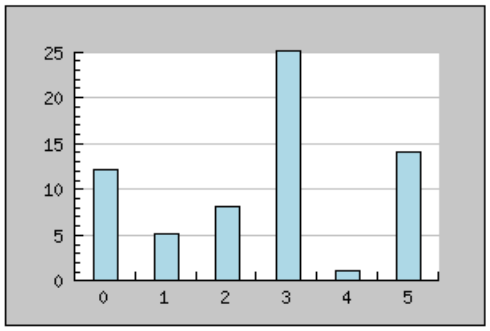


FIG. 4.7 – Diagramme de barres

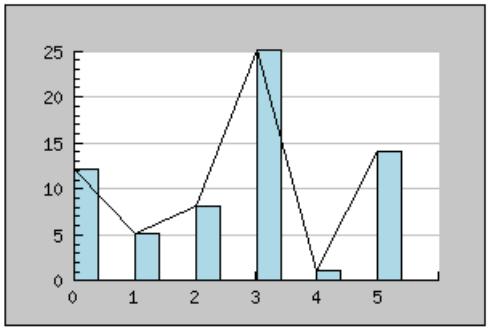


FIG. 4.8 – Diagramme de lignes/barres

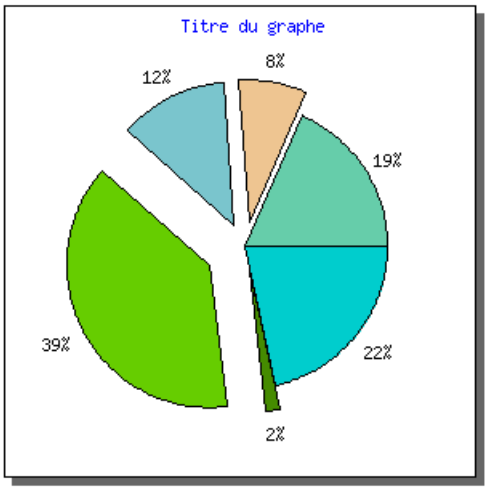


FIG. 4.9 – Diagramme camembert

```
  // Création de l'objet graphique
  $graph = new PieGraph(300,300);
  // Défini une ombre sous le graphe
  $graph->SetShadow();
  // Définition du titre du graphe
  $graph->title->Set("Titre du graphe");
  $graph->title->SetColor("blue");
  // Creation du "Pie Plot"
  $pie = new PiePlot($data);
  // Choix des couleurs du diagramme
  $pie->SetTheme("earth");
  // Défini le nbre de pixels par tranches
  $pie->Explode(array(0,15,15,25,15));
  // Ajout au graphe
  $graph->Add($pie);
  // Affichage du graphe
  $graph->Stroke();
?>
```

Nous pouvons voir ici d'autres méthodes (utilisable également avec les diagrammes précédents) : la méthode `SetShadow()` qui ajoute une ombre sous le graphe, les méthodes `title->Set()` et `title->SetColor()` qui définissent le titre et sa couleur, et enfin `PiePlot()` qui crée un diagramme en camembert, `SetTheme()` qui défini sa couleur et `Explode()` qui défini la taille des tranches. Le résultat de ce script est visible en figure 4.9.

4.2.2 Conclusion

Cette librairie permet donc de générer très facilement et très rapidement des diagrammes de type très différents. Cet outil est très utile pour représenter des statistiques effectuées sur une base MySQL par exemple.

4.3 Les extensions PHP

On distingue deux projets : PEAR pour "**P**HP **E**xtension and **A**pplication **R**epository" (`http://pear.php.net`) et PECL pour "**P**HP **E**xtension **C**ommunity **L**ibrary" (`http://pecl.php.net`).

4.3.1 PEAR

Tout d'abord PEAR est également le nom du programme permettant de récupérer les paquetages PEAR sur internet et de les installer. Pour récupérer ce programme : `apt-get install pear` puis, pour chaque paquetage à installer (visible sur le site de PEAR) : `pear install paquetage`.
Les paquetages (packages, ou modules) de PEAR sont définis de manière très précise en terme de projet, équipe de développement, numéro de version, cycle de publication, documentation et dépendances vis-à-vis d'autres paquetages. Les paquetages peuvent être installés grâce au logiciel PEAR (fortement recommandé), soit à partir d'une archive tar compressée.
Il existe deux types de paquetages : les paquetages "source", qui ne contiennent que des fichiers de code source, et les paquetages "binaires", qui contiennent des fichiers binaires et demandent un environnement de compilation.
L'utilisation du programme PEAR est très simple :

- `pear list` permet d'afficher la liste des paquetages PEAR installés.
- `pear remote-list` affiche la liste des paquetages disponibles.
- `pear upgrade-all` mise à jour de tous les paquetages installés.
- `pear upgrade pack` mise à jour du paquetage "pack".
- `pear install pack` installe le paquetage "pack".

4.3.2 PECL

PECL etait jadis un sous-ensemble de PEAR pour les extensions de PHP écrites en C. Les extensions dans PECL suivent la convention de codage PHP (et non plus celle de PEAR), mais elles sont distribuées et installées comme des paquetages PEAR.

5

XML

Un Anneau pour les gouverner tous.
Un Anneau pour les trouver,
Un Anneau pour les amener tous
et dans les ténèbres les lier.
J. R. R. Tolkien

XML (eXtensible Markup Language) est un méta-langage[11], indépendant des plate-formes et des systèmes d'exploitation, permettant de décrire la structure hiérarchique d'un document. Je présenterai ici rapidement ce méta-langage, les DTD (Document Type Definition) qui permettent de valider un document par rapport à un méta-modèle et les feuilles de style XSLT .

5.1 Structure de document XML

Un document XML est constitué de tags comme en HTML. Mais ces tags ne sont pas statiques : ils sont définis par l'utilisateur. Seuls les mots commençant par XML ou xml sont réservés. Exemple :

```
<VAISSEAU>
  <MODELE>X-WING</MODELE>
  <ARMEMENT>
    <PUISSANCE_FEU>4</PUISSANCE_FEU>
    <CANON_PROTON>OUI</CANON_PROTON>
  </ARMEMENT>
</VAISSEAU>
```

[11] Langage permettant de définir les règles et les symboles d'un autre langage.

Un commentaire s'écrit comme en HTML : `<!-- ... -->`.
Les instructions spéciales pour les applications sont comprises entre `<?` et `?>`. Et les instructions pour les règles sont comprises entre `<! ... >`.

Un document XML se compose de :
- Un prologue, qui permet de préciser qu'il s'agit d'un document XML, et d'identifier le jeu de carctères utilisé ainsi que la grammaire (DTD).
- Un arbre d'éléments (le document) contenant un seul élément racine[12].
- Des commentaires et des instructions de traitement.

Un document sera bien formé s'il respecte la syntaxe XML et valide s'il vérifie la DTD.

5.1.1 Prologue

Si aucun jeu de caractères n'est spécifié en début de document, toute application XML supposera qu'il s'agit du jeu de caractère Unicode encodé en UTF-8 ou UTF-16. La première ligne d'un document XML respectant un encodage UTF-8 serait donc :

```
<?xml version="1.0" encoding="UTF-8"?>
```

Il faut ensuite déclarer la grammaire. Elle se situe en général dans un autre fichier. On y fait référence à l'aide d'une balise `<!DOCTYPE elt_racine url_DTD>`. *elt_racine* définit l'élément racine du document (`VAISSEAU` dans l'exemple précédent), et *url_DTD* indique l'URL où se situe la DTD.
Il est possible de déclarer la DTD directement dans le document par :

```
<!DOCTYPE elt_racine
  [
    ... déclaration de la DTD ...
  ]
>
```

Dans la déclaration d'une DTD, on utilise des entités et des éléments.

5.1.1.1 Entités

Les entités peuvent être assimilées à des alias : elles vont remplacer un nom par un autre. On distingue deux types d'entités : les entités générales `<!ENTITY nom_a_remplacer nom_remplacement>` et les entités paramètre `<!ENTITY % nom_a_remplacer nom_remplacement>`. Ainsi, par exemple, avec la déclaration :

```
<!ENTITY LCB "Laboratoire de Chimie Bactérienne">
```

Chaque occurrence de `&LCB;` dans le document sera remplacée par "Laboratoire de Chimie Bactérienne". De même, avec :

```
<!ENTITY % entier "INTEGER">
```

Chaque occurrence de `%entier;` sera remplacée par "INTEGER".
Un type d'entité un peu plus particulier est également disponible : les entités externes qui permettent d'utiliser des documents externes comme éléments de remplacement grâce au mot clé

[12] Un élément racine est un élément contenant tous les autres.

SYSTEM :
<!ENTITY fichier SYSTEM "http://cmi.univ-mrs.fr/traduction.txt">
Chaque occurrence de &fichier; sera remplacé par le contenu du document *traduction.txt*. Et enfin, une entité peut être remplacé par l'un (ou plusieurs) des éléments d'une liste (en suivant les règles des expressions régulières) :
<!ENTITY % Vaisseau (X-Wing | Y-Wing | A-Wing)*>
%Vaisseau; sera remplacé par une association quelconque[13] des termes X-Wing, Y-Wing, ou A-Wing.

5.1.1.2 Elements

Le mot clé <!ELEMENT nom contenu> permet de définir un contenu pour un élément donné, c'est-à-dire la hiérarchie à laquelle il va obéir ou la grammaire proprement dite. Prenons un exemple simple tiré de la grammaire HTML :
<!ELEMENT HTML (HEAD, BODY)>
Ceci signifie que l'élément HTML contient un élément HEAD suivi d'un élément BODY, soit :

```
<HTML>
  <HEAD>
    ...
  </HEAD>
  <BODY>
    ...
  </BODY>
</HTML>
```

A un objet, on peut adjoindre un ou des attributs par la directive <!ATTLIST> :

```
<!ELEMENT BODY (P | DIV)*>
<!ATTLIST BODY
   bgcolor (white | black) #REQUIRED
  align CDATA "center">
```

On a par exemple défini pour BODY un attribut *bgcolor* obligatoire (#REQUIRED), et un autre (*align*) optionnel (mais de type CDATA c'est-à-dire "chaîne de caractères") avec pour valeur par défaut *center*.
Enfin, le mot clé EMPTY définit un élément sans contenu (comme
 par exemple), et (#PCDATA) défini d'autres éléments de marquage ou un autre contenu quelconque (chaîne de caractères par exemple).

5.1.2 Corps du document

Il s'agit de l'arbre hiérarchique des éléments. Il existe toujours un et un seul élément supérieur qui contient tous les autres. Chaque élément est défini par :
<nom>contenu de l'élément</nom>
On peut ajouter qu'un élément peut contenir d'autres éléments ou du texte, et se situe à

[13] Ou aucun à cause de *. Pour en avoir au moins un il aurait fallu remplacer * par +.

l'intérieur d'un seul élément.

5.1.2.1 Exemple de DTD

Nous pouvons maintenant créer une petite DTD d'exemple. Il s'agira d'un annuaire :
annuaire.dtd :

```
<!ELEMENT annuaire (adresse+)>

<!ELEMENT adresse (prenom, nom, rue+, ville, pays?, codepostal?)>
<!ATTLIST adresse type (domicile|bureau) #REQUIRED>

<!ELEMENT prenom (premier, deuxieme, troisieme?)>
<!ELEMENT premier (#PCDATA)>
<!ELEMENT deuxieme (#PCDATA)>
<!ELEMENT troisieme (#PCDATA)>

<!ELEMENT nom (#PCDATA)>

<!ELEMENT rue (#PCDATA)>

<!ELEMENT ville (#PCDATA)>

<!ELEMENT pays (#PCDATA)>

<!ELEMENT codepostal (#PCDATA)>
<!ATTLIST codepostal longueur CDATA "6">
```

On définit un élément racine qui est `annuaire` et qui contient un élément `adresse`, un élément `prenom`, un élément `nom`, un ou plusieurs éléments `rue`, un élément `ville`, deux éléments optionnels `pays` et `codepostal`.
`adresse` a un attribut obligatoire qui vaut "domicile" ou "bureau".
`prenom` contient trois éléments dont le dernier est optionnel. Ces champs sont tous de type chaîne de caractères.
Les éléments `nom`, `rue`, `ville`, `pays`, et `codepostal` sont de type chaîne de caractères.
Enfin, `codepostal` a un attribut longueur de type chaîne de caractère et qui a pour valeur par défaut "6". Pour utiliser cette DTD dans un fichier XML, nous ferons alors :

annuaire.xml

```
<?xml version="1.0" encoding="ISO-8859-1">
<!DOCTYPE annuaire SYSTEM "annuaire.dtd">
<annuaire>

  <adresse type="domicile">
    <prenom>
      <premier>Pierre</premier>
      <deuxieme>Jean</deuxieme>
      <troisieme>Paul</troisieme>
    </prenom>
    <nom>Marcel</nom>
    <rue>14, rue du Cyprès bleu</rue>
    <rue>Près du Chêne vert</rue>
    <ville>Bouzilly-sur-Oise</ville>
    <codepostal longueur="5">75565</codepostal>
  </adresse>

</annuaire>
```

5.2 Les feuilles de style XSLT

XSLT (eXtended Stylesheet Language Transformation) est un langage de transformation d'un document source XML vers un autre document XML[14]. Il nécessite un processeur qui n'est pas inclus dans les distributions PHP standard : *Sablotron*, téléchargeable par
`apt-get install sablotron`[15].
Un programme XSLT est un document XML dont l'élément racine est `<xsl:stylesheet>`. Ou plus précisément, en spécifiant la recommandation utilisée :
`<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`
Les instructions XSLT sont toutes préfixées par "`xsl:`". Les objets les plus importants de ce modèle sont les règles – ou templates – `<xsl:template>`. Elles s'appliquent à des éléments spécifiques du fichier XML grâce à la méthode `match`. Par exemple, pour être appliquée à l'ensemble du document on utilisera :
`<xsl:template match="/">`
Pour une application à l'élément `<prenom>` :
`<xsl:template match="prenom">`
Et pour une application directe à l'élément `<prenom>` fils de `<adresse>`
(`<adresse><prenom></prenom></adresse>`) :
`<xsl:template match="adresse/prenom">`
Le format de sortie du document traduit est spécifié par `<xsl:ouput>` qui admet notamment

[14] Dont le XHTML, SVG, ...
[15] Le nom de l'exécutable est "sabcmd".

les attributs *method* pour le type de sortie (XML, XHTML, TEXT) et *encoding* pour le jeu de caractères utilisé. Pour une sortie classique en XHTML, on utilisera alors :
`<xsl:ouput method="html" encoding="ISO-8859-1">`
Pour résumer, un programme XSLT s'applique à un document XML et génère un autre document XML dans lequel toute trace d'instruction XSL aura disparu. Prenons l'exemple du fichier XML précédent (*annuaire.xml*) : nous voulons maintenant présenter les résultats sous forme de page web. Nous allons donc créer un programme xslt qui définira la règle de production générale du document, puis, un programme définissant les règles utilisées pour chacun des blocs de notre fichier XML.
presentation_generale.xsl :

```
<?xml version="1.0" encoding="ISO-8859-1">

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:include href="adresse.xsl" />
  <xsl:output method="html" encoding="ISO-8859-1" />
  <xsl:template match="/">
    <!-- Entete de la page HTML -->
    <HTML>
      <HEAD>
        <TITLE>Annuaire avec affichage formaté par XSLT</TITLE>
      </HEAD>
      <BODY>
        <CENTER><H1>Affichage de l'annuaire mis en forme par XSLT</H1><CENTER>
        Liste des individus référencés dans notre annuaire :

        <!-- Lecture du document XML -->
        <xsl:apply-templates />

      </BODY>
    </HTML>
  </xsl:template>

</xsl:stylesheet>
```

Nous avons défini un document XML (ligne `<?xml>`) qui est qui plus est un document XSLT (ligne `<xsl:stylesheet>`). Nous avons inclus un autre fichier "prenom.xsl" (voir définition ci-après) contenant des règles (`<xsl:include>`). Le format de sortie choisie est le XHTML (`<xsl:output>`). Ensuite, toutes les lignes situées entre les balises `<xsl:template>` seront appliquées à l'ensemble du document XML sur lequel on appliquera le fichier "presentation_generale.xsl". Les lignes qui ne sont pas des commandes XSLT (en l'occurrence le code HTML) seront retranscrites sans modification. Puis, les règles de modifications (contenues dans le fichier "prenom.xsl") sont appliquées (`<xsl:apply-templates>`). Etudions maintenant

plus précisément ces règles.
prenom.xsl :

```
<?xml version="1.0" encoding="ISO-8859-1">

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="adresse">
    <B>
      <xsl:value-of select="nom" /> <xsl:value-of select="prenom@premier" /><BR />
    </B>
    <xsl:choose>
      <xsl:when test="adresse@type='domicile'">
        Adresse PRIVE
      </xsl:when>
      <xsl:otherwise>
        Adresse PROFESSIONNELLE
      </xsl:otherwise>
    </xsl:choose>
    <BR />
    <xsl:for-each select="rue">
      <I><xsl:value-of select="rue" /></I><BR />
    </xsl:for-each>
    ...
  </xsl:template>

</xsl:stylesheet>
```

Dans cet exemple nous voyons apparaître de nouvelles instructions XSLT :

- `<xsl:value-of select="nom_elt" />` : cette balise est remplacée par le contenu de la balise XML de nom *nom_elt*. Par exemple, avec un fichier XML contenant `<nom>Marcel</nom>`, la ligne `<xsl:value-of select="nom">` sera remplacée par "Marcel".
- `<xsl:choose>`, `<xsl:when test="condition">`, et `<xsl:otherwise>` : effectue des actions en fonction de conditions.
- `<xsl:for-each select="nom_elt">` :opère une itération sur tous les éléments *nom_elt* et effectue la même opération pour chacun de ces éléments.

En appliquant cette feuille de style XSLT "presentation_generale.xslt" à notre fichier "annuaire.xml", nous obtiendrions la sortie :

```
<HTML>
  <HEAD>
    <TITLE>Annuaire avec affichage formaté par XSLT</TITLE>
  </HEAD>
  <BODY>
    <CENTER><H1>Affichage de l'annuaire mis en forme par XSLT</H1><CENTER>
    Liste des individus référencés dans notre annuaire :
    <B>
      Marcel Pierre<BR />
    </B>
    Adresse PRIVE
    <BR />
    <I>14, rue du Cyprès bleu</I><BR />
    <I>Près du Chêne vert</I><BR />
  </BODY>
</HTML>
```

5.3 DOM

DOM (Document Object Model) est une API (interface de programmation d'applications) pour les documents HTML et XML. Ce modèle est indépendant de tout langage de programmation (et pourra donc être implémenté dans n'importe quel langage). Il définit la structure logique des documents et la manière par laquelle on va accéder et manipuler un document en construisant l'arbre hiérarchique. Il est en général utilisé entre le parseur XML et une application de plus haut niveau, mais est très gourmand en mémoire et il est de plus très lent.

5.4 SAX

SAX (Simple API for XML) est, comme son nom l'indique, une API pour XML. Il s'agit d'un travail effectué par une communauté de développeur de parseurs : ces spécifications ne sont donc pas gérées par le W3C. Contrairement à DOM il s'agit d'une approche évenementielle et d'une lecture séquentielle du document XML (notion de début et de fin de document). Elle permet de récupérer les informations lues par le parseur XML sans implémenter d'arbre en mémoire : elle est donc particulièrement adaptée au traitement de gros documents XML.

5.5 Utiliser les documents XML avec PHP

Je distinguerai deux cas d'utilisation de documents XML avec PHP : l'utilisation avec des commandes de PHP4 et l'utilisation avec des commandes de PHP5 (les instructions PHP4 restant valables). Je m'appuierai sur ici une API SAX (voir l'extension DOM XML de PHP pour programmer avec l'API DOM).

5.5.1 SAX

L'analyse d'un document XML s'effectue en trois étapes :
- On initialise un parseur grâce à la fonction `xml_parser_create()`.
- Suivant les différents types de balises détectées dans le document on indique les actions qui doivent être effectuées par le parseur.
- Enfin, on lance l'analyse du document avec `xml_parse()`.

Voici un exemple très simple de parseur permettant la détection des balises ouvrantes et fermantes :
parse_xml.php :

```
<?php

  function DebutBalise($parser, $nom, $attributs)
  {
    print "Balise ouvrante $nom\n";
    if (!empty($attributs))
    {
      print "  attribut(s) :\n";
      while (list($cle, $val)=each($attributs))
        print "    $cle = $val\n";
    }
  }

  function FinBalise($parser, $nom)
  {
    print "Balise fermante $nom\n";
  }

  function InfoTextuelle($parser, $chaine)
  {
    if (trim($chaine)!="")
    print "  Noeud : $chaine\n";
  }

  header("Content-type: text/plain");

  $parseur=xml_parser_create();
  xml_set_element_handler($parseur, "DebutBalise", "FinBalise");
  xml_set_character_data_handler($parseur, "InfoTextuelle");

  if (!($file=fopen("annuaire.xml", "r")))
  {
    print "Impossible d'ouvrir le fichier annuaire.xml";
```

```
    exit;
  }

  while ($data=fread($file, 1024))
  {
    if (!xml_parse($parseur, $data, feof($file)))
    {
      sprintf("XML error: %s at line %d",
        xml_error_string(xml_get_error_code($parseur)),
        xml_get_current_line_number($parseur));
      exit;
    }
  }

  fclose($file);
  xml_parser_free($parseur);

?>
```

On distingue bien ici la création du parseur, puis la définition des actions effectuées par les fonctions "DebutBalise", "FinBalise", et "InfoTextuelle" et appelées par les fonctions `xml_set_element_handler(nom_parseur, nom_fct_debut_balise, nom_fct_fin_balise)` et `xml_set_character_data_handler(nom_parseur, nom_fct_noeud)`. Enfin, on affiche les erreurs éventuelles renvoyées par le parseur `xml_parse(nom_parseur, donnees, final)`[16] dans `xml_get_error_code(parseur)` (retour du code d'erreur), `xml_error_string(code_erreur)` (traduction du code d'erreur en chaîne de caractères) et `xml_get_current_line_number(parseur)` (donne la ligne courante du parseur).

5.5.2 SimpleXML

Pour terminer, voici un aperçu des nouveautés apportées par l'extension SimpleXML de PHP5. Pour parser un fichier XML comme précédemment, on utilisera tout simplement la commande `simplexml_load_file()` qui crée un objet de type SimpleXMLElement :
`$objetXML=simple_xml_load_file("annuaire.xml");`
Il est maintenant très simple d'accéder aux données : pour accéder au premier prénom par exemple, on utilisera la variable `$objetXML->prenom->premier`. On accède aux attributs d'un élément par la méthode `attributes()`. `$objetXML->codepostal->attributes()` renvoie une table de hachage contenant le nom et la valeur des attributs de l'élément "codepostal".
Cette extension permet donc d'effectuer très facilement, en très peu de lignes, les commandes XML les plus simples (d'où le nom de cette extension ...).

[16] Les fragments XML lus peuvent etre incomplets tant que final à eu valeur booléenne Fausse.

6

La sécurité

Le concept de l'opensource oblige distributeurs et développeurs à rester honnêtes
Linus Torvalds

JE présenterai rapidement dans ce chapitre les différentes techniques permettant de sécuriser l'accès web à des sites (notamment des sites en PHP/MySQL bien sûr).

6.1 Le protocole https

Le protocole http est utilisé pour effectuer des accès à des pages web en empruntant le réseau internet. Les requêtes et réponses envoyées et faites par le serveur peuvent être interceptées par un pirate. Le protocole https (pour http sécurisé) permet d'encapsuler et de crypter le trafic http : un pirate interceptant ces données devra donc en plus les déchiffrer pour pouvoir les utiliser. De plus, le protocole https certifie que le serveur auquel on pense accéder est bien celui auquel on croit (pas de redirection masquée).
Le fonctionnement est classique : les données sont cryptées et décryptées à l'aide d'un couple de clés (système de cryptographie asymétrique RSA) :

- la clé publique que tout le monde connaît
- la clé privée qui n'est connue que du serveur

Si A veut envoyer un message à B, il crypte son message avec la clé publique de B et B pourra décrypter le message grâce à sa clé privée.
La clé publique du serveur doit être authentifiée par un organisme, l'Autorité de certification, qui délivre un certificat (sinon le navigateur affichera des messages de mise en garde). Ces certificats sont connus du navigateur ou alors, on peut les accepter comme valides si l'on est sûr de la provenance des informations.

6.2 Serveur SSH

SSH signifie *Secure SHell*. C'est un protocole qui permet de faire des connexions sécurisées (cryptées) entre un serveur et un client SSH. Comme nous l'avons vu, le trafic de données sur internet peut être intercepté, d'où la nécessité de crypter les données sensibles (mots de passe, ...). Pour installer un serveur SSH, nous aurons recours au traditionnel `apt-get install ssh`. SSH s'appuie sur des algorithmes à paire de clés : une clé publique et une clé privé. L'identification peut se faire lors de la connexion à l'aide d'un mot de passe ou directement grâce aux clés. Une connexion typique SSH se fait par :
`ssh -p num_port -l login machine`
ce qui permet à l'utilisateur identifié par *login* de se connecter au port *num_port* du serveur *machine*. Pour des explications plus précises sur la configuration d'un serveur SSH et de l'authentification automatique, je vous renvoie aux explications du site `http://www.lea-linux.org`. Le SSH permet, comme le `telnet` le `Xforwarding` (utilisation en mode graphique d'un logiciel présent sur une machine distante). Pour celà, il faut activer l'option `X11Forwarding` dans les fichiers de configuration (`/etc/ssh/sshd_config` et `/etc/ssh/ssh_config`). La commande typique est alors :
`ssh login@machine programme`
qui connecte l'utilisateur *login* sur *machine* et exécute *programme*.
On peut également effectuer des copies de fichiers sécurisées grâce à la commande `scp` – pour SecureCoPy :
`scp source:fichier destination:fichier_cible`
permet de copier *fichier* depuis la machine *source* vers la machine *destination* dans le fichier *fichier_cible*.
Enfin, il est également possible de transférer des fichiers par l'intermédiaire d'un ftp sécurisé : le sftp – pour SecureFTP. Les commandes sont identiques à celle d'un ftp normal et certains utilitaire de ftp graphique gèrent le sftp (comme gFTP par exemple).

6.3 Protection de pages par authentification au niveau du serveur

Si l'on dispose de la maîtrise totale ou partielle de la configuration du serveur HTTP (en général Apache), il est possible de protéger des pages d'un site à l'aide d'un mécanisme d'authentification par nom et mot de passe. Ce mécanisme utilise directement une fonctionnalité du protocole et c'est le navigateur client qui prendra en charge l'aspect graphique et la récupération des données. Il suffit donc de spécifier quelles sont les pages à protéger et quels sont les utilisateurs autorisés à y accéder.
Le fichier de configuration de cette protection doit se nommer `.htaccess` et doit être placé dans le répertoire à protéger. Voici un fichier `.htaccess` classique :

```
AuthType digest
AuthName "Zone securisée"
AuthUserFile /etc/apache/users
Require valid-user
```

FIG. 6.10 – Aperçu de la fenêtre d'authentification.

Les champs sont donc les suivants :

- `AuthType` : type d'authentification (basic ou digest). Avec digest les login et mot de passe sont cryptés sur le réseau (mais il peut y avoir des problèmes de compatibilité). Une authentification basique – basic – fonctionnera, elle, dans tous les cas.
- `AuthName` : indique le nom du schéma d'autorisation. Ce nom sera transmis au navigateur du client afin qu'il sache quels login et mot de passe retourner au serveur.
- `AuthUserFile` : nom du fichier contenant les identifiants de connexion. Ce fichier devra être créé à l'aide de la fonction `htpasswd` qui va créer un fichier texte contenant les couples login :mot de passe (crypté). Cette commande admet les options `-c` pour créer un nouveau fichier et `-b` pour prendre le mot de passe sur la ligne de commande plutôt qu'interactivement. Exemple :
  `htpasswd -cb /etc/apache/users login password`
  Pour ajouter un nouvel utilisateur il suffira de réexécuter cette commande en ommettant l'option `-c` ; pour supprimer un utilisateur il faudra éditer le fichier et supprimer manuellement la ligne.
- `Require` : permet de définir qui a accès au contenu. `valid-user` indique que n'importe quel utilisateur de la liste `AuthUserFile` peut avoir accès au contenu s'il s'identifie correctement. Sinon, on peut spécifier ici la liste exhaustive des utilisateurs autorisés.

Enfin, pour activer la protection, il faut s'assurer que dans le fichier de configuration d'Apache `/etc/appache/httpd.conf`, le paramètre `AllowOverride` est différent de `None`. On peut lui donner la valeur `All` (sécurité insuffisante) ou `AuthConfig` (c'est la valeur que nous utiliserons ici).
Ainsi, lorsque l'on voudra afficher une page protégée par ce système, le navigateur affichera une fenêtre du même type que celle présentée en figure 6.10.

6.4 Protection des accès MySQL depuis PHP

La définition des paramètres de connexion à la base MySQL (HOST, USER, PASSWORD) stockés dans un fichier extérieur (en utilisant des `define()` au script PHP peuvent être incorporés au moyen d'une directive `include()`. Si de plus ce fichier est placé en dehors de l'arborescence des répertoires consultés normalement par apache (`/var/www`) et protégé par un fichier `.htaccess`, alors la protection sera maximale.

6.5 Protection des données de formulaires

Lors de la visualisation du code source, la valeur des champs "hidden" peut être lue. Pour remédier à celà on peut utiliser les fonctions `base64_encode()` pour coder la valeur de la variable et `base64_decode()` pour décoder la valeur.

6.6 Sécuriser les cookies avec une clé cryptée

Les cookies étant des fichiers texte sur l'ordinateur du client, les données qu'il contient peuvent être aisément corrompues. Pour éviter cela, on peut ajouter une clé cryptée. Par exemple il peut s'agir de la concaténation de toutes les valeurs cryptées en utilisant `crypt()` ou `base64_encode()`, la somme des valeurs cryptées, ...

6.7 Méthodes de cryptage

Pour une fonction de cryptage plus efficace et paramétrable (admet de multiples algorithmes), voir `mcrypt_encode()` et `mcrypt_decode()`.

7

Un exemple concret : le site du LCB

I think Linux is going to be big, but, hey, who am I to say ?
Linus Torvalds

VOICI un exemple de mise en application de PHP/MySQL pour l'administration du site internet du Laboratoire de Chimie Bactérienne (CNRS) : `http://www.lcb.cnrs-mrs.fr`. Les pages présentées dans la suite ne sont pas accessibles depuis le site car protégées par un fichier `.htaccess` (normal sinon n'importe qui pourrait modifier le site). Cette partie administration prend en charge la gestion des actualités du laboratoire, de l'annuaire du personnel et des publications.

7.1 Les actualités

Les actualités sont gérées dans une table `ACTUS` de format :

Actus	
ID	INT
DATE	VARCHAR(11)
DESCRIPTION	TEXT

Une interface permet la saisie/modification/suppression des actualités (voir figure 7.11) qui seront par la suite récupérées dans la base et affichées automatiquement dans la page "Actualités" disponible pour tous les utilisateurs.

Voici la structure générale du fichier `actus.php`. On remarquera que les parties du site apparaissant dans de nombreuses pages (menu, etc.) sont incluses grâce à des fonctions PHP paramétrables (`header.php`, `menu.php`). Il en va de même pour les fonctions courantes d'accès à la base (`Connexion.php`, `Param_connexion.php`, `Requetes.php`).

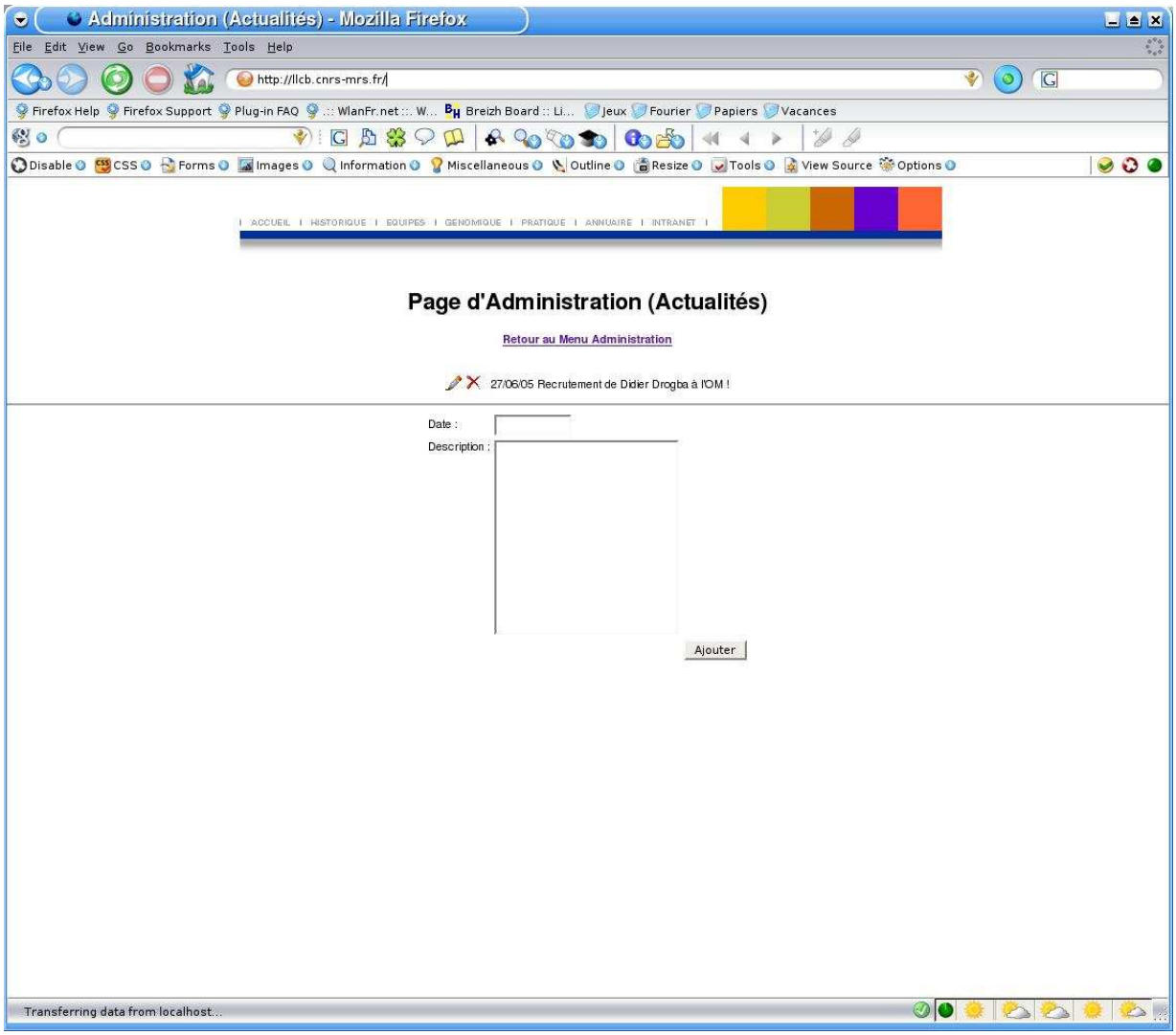


Fig. 7.11 – Ecran de saisie des actualités

```
<?PHP
    include("../header.php");
    head("Administration (Actualités)");
?>

    <BODY style="margin: 0px">

        <H2 id="Debut"></H2>
        <TABLE width="100%" border="0" cellspacing="0" cellpadding="0">
            <TR>
                <TD align="center">
                    <?PHP
                        include("../menu.php");
                        barre_menu("");
                    ?>
                </TD>
            </TR>
        </TABLE>
        <DIV align="center">
            <H1>
                Page d'Administration (Actualités)
            </H1>
            <H2>
                <A href="admin.php">Retour au Menu Administration</A>
            </H2>
            <BR />
            <?PHP
                require_once("../Connexion.php");
                require_once("Param_connexion.php");
                require_once("../Requetes.php");
                $connexion=Connexion(NOM, PASSE, BASE, SERVEUR);
                if (!empty($_POST{"submit"}))
                {
                    // Insertion d'une nouvelle news
                    $ident=determine_ID("LCB.ACTUS", $connexion);
                    $requete="INSERT INTO LCB.ACTUS (ID, DATE, DESCRIPTION) VALUES
                              (\"$ident\", \"".$_POST{"Date"}."\", \"".
                              $_POST{"Description"}."\")";
                    $resultat=ExecRequete($requete, $connexion);
                }
                if (!empty($_POST{"modif"}))
                {
                    // Modification d'une news dans la base
                    $resultat=ExecRequete("UPDATE LCB.ACTUS SET DATE=\"".
```

```
                                $_POST{"Date"}."\", DESCRIPTION=\"".
                                $_POST{"Description"}."\" WHERE ID=\"".
                                $_POST{"ID"}."\"", $connexion);
}
if (!empty($_GET{"mode"}))
{
    if ($_GET{"mode"}==1)
    {
        // Modifier une news
        $resultat=ExecRequete("SELECT * FROM LCB.ACTUS WHERE ID=".
                                $_GET{"elt"}, $connexion);
        $elt=mysql_fetch_object($resultat);
        print "<FORM method=\"POST\" action=\"actus.php\">\n";
        print "<TABLE>\n";
        print "<TR>\n";
        print "<TD>Date :</TD><TD><INPUT type=\"text\" size=\"10\"
               name=\"Date\" value=\"".$elt->DATE."\"/></TD>\n";
        print "</TR>\n";
        print "<TR>\n";
        print "<TD vAlign=\"top\">Description :</TD><TD><TEXTAREA
               name=\"Description\" cols=\"20\" rows=\"10\">".
               $elt->DESCRIPTION."</TEXTAREA></TD>\n";
        print "</TR>\n";
        print "<TR>\n";
        print "<TD></TD><TD></TD><TD><INPUT type=\"submit\" name=\"modif\"
               value=\"Modifier\" /></TD>\n";
        print "</TR>\n";
        print "</TABLE>\n";
        print "<INPUT type=\"hidden\" name=\"ID\" value=\"".
               $_GET{"elt"}."\" />\n";
        print "</FORM>\n";
    }
    else
    {
        // Supprimer une news
        $resultat=ExecRequete("DELETE FROM LCB.ACTUS WHERE ID=".
                                $_GET{"elt"}, $connexion);
    }
}
if ($_GET{"mode"}!=1)
{
    $resultat=ExecRequete("SELECT * FROM LCB.ACTUS ORDER BY ID",
                                $connexion);
    print "<TABLE>\n";
```

```
                    while ($elt=mysql_fetch_object($resultat))
                    {
                        print "<TR>\n";
                        print "<TD><A href=\"actus.php?mode=1&amp;elt=".$elt->ID.
                               "\"><IMG src=\"images/modif.png\" alt=\"Modif\"
                               border=\"0\"/></A></TD><TD>
                               <A href=\"actus.php?mode=2&amp;elt=".$elt->ID.
                               "\"><IMG src=\"images/supp.png\" alt=\"Supp\"
                               border=\"0\"/></A><TD></TD><TD>".
                               $elt->DATE."</TD><TD>".$elt->DESCRIPTION."</TD>\n";
                        print "</TR>\n";
                    }
                    print "</TABLE>\n";
                    print "<HR />\n";
                    print "<FORM method=\"POST\" action=\"actus.php\">\n";
                    print "<TABLE>\n";
                    print "<TR>\n";
                    print "<TD>Date :</TD><TD><INPUT type=\"text\" size=\"10\"
                           name=\"Date\" /></TD>\n";
                    print "</TR>\n";
                    print "<TR>\n";
                    print "<TD vAlign=\"top\">Description :</TD><TD><TEXTAREA
                           name=\"Description\" cols=\"20\" rows=\"10\">
                           </TEXTAREA></TD>\n";
                    print "</TR>\n";
                    print "<TR>\n";
                    print "<TD></TD><TD></TD><TD><INPUT type=\"submit\"
                           name=\"submit\" value=\"Ajouter\" /></TD>\n";
                    print "</TR>\n";
                    print "</TABLE>\n";
                    print "</FORM>\n";
                }
            ?>
        </DIV>
    </BODY>

</HTML>
```

7.2 L'annuaire

Pour l'annuaire, on utilise le même type de gestion : stockage des informations reltives au personnel dans une table de données avec une page sécurisée permettant la gestion de cette table (figure 7.13). Les éléments du formulaire pour lesquels une partie de l'information est

connue ont été pré-remplis (par exemple, pour l'adresse mail, on sait que tout le monde aura pour nom de domaine : `ibsm.cnrs-mrs.fr`).

Le code correspondant à cette partie (`annuaire.php`) est le suivant :

```
<?PHP
    include("../header.php");
    head("Administration (Annuaire)");
    include("../body_load_images.php");
?>

        <H2 id="Debut"></H2>
        <TABLE width="100%" border="0" cellspacing="0" cellpadding="0">
            <TR>
                <TD align="center">
                    <?PHP
                        include("../menu.php");
                        barre_menu("");
                    ?>
                </TD>
            </TR>
        </TABLE>
        <DIV align="center">
            <H1>
                Page d'Administration (Annuaire)
            </H1>
            <H2>
                <A href="admin.php">Retour au menu Administration</A>
            </H2>
            <BR />
            <?PHP
                require_once("../Connexion.php");
                require_once("Param_connexion.php");
                require_once("../Requetes.php");
                $connexion=Connexion(NOM, PASSE, BASE, SERVEUR);
                if (!empty($_POST{"Modification"}))
                {
                    // Modification effective d'un individu
                    $requete="UPDATE LCB.ANNUAIRE SET PRENOM=\"".$_POST{"Prenom"}."\",
                              NOM=\"".$_POST{"Nom"}."\", TITRE=\"".$_POST{"Titre"}."\",
                              GRADE=\"".$_POST{"Grade"}."\", MAIL=\"".$_POST{"Mail"}.
                              "\", AFFECTATION=\"".$_POST{"Affectation"}."\",
                              EQUIPE=\"".$_POST{"Equipe"}."\", TEL=\"".$_POST{"Tel"}.
                              "\" WHERE ID=\"".$_POST{"ID"}."\"";
                    ExecRequete($requete, $connexion);
```

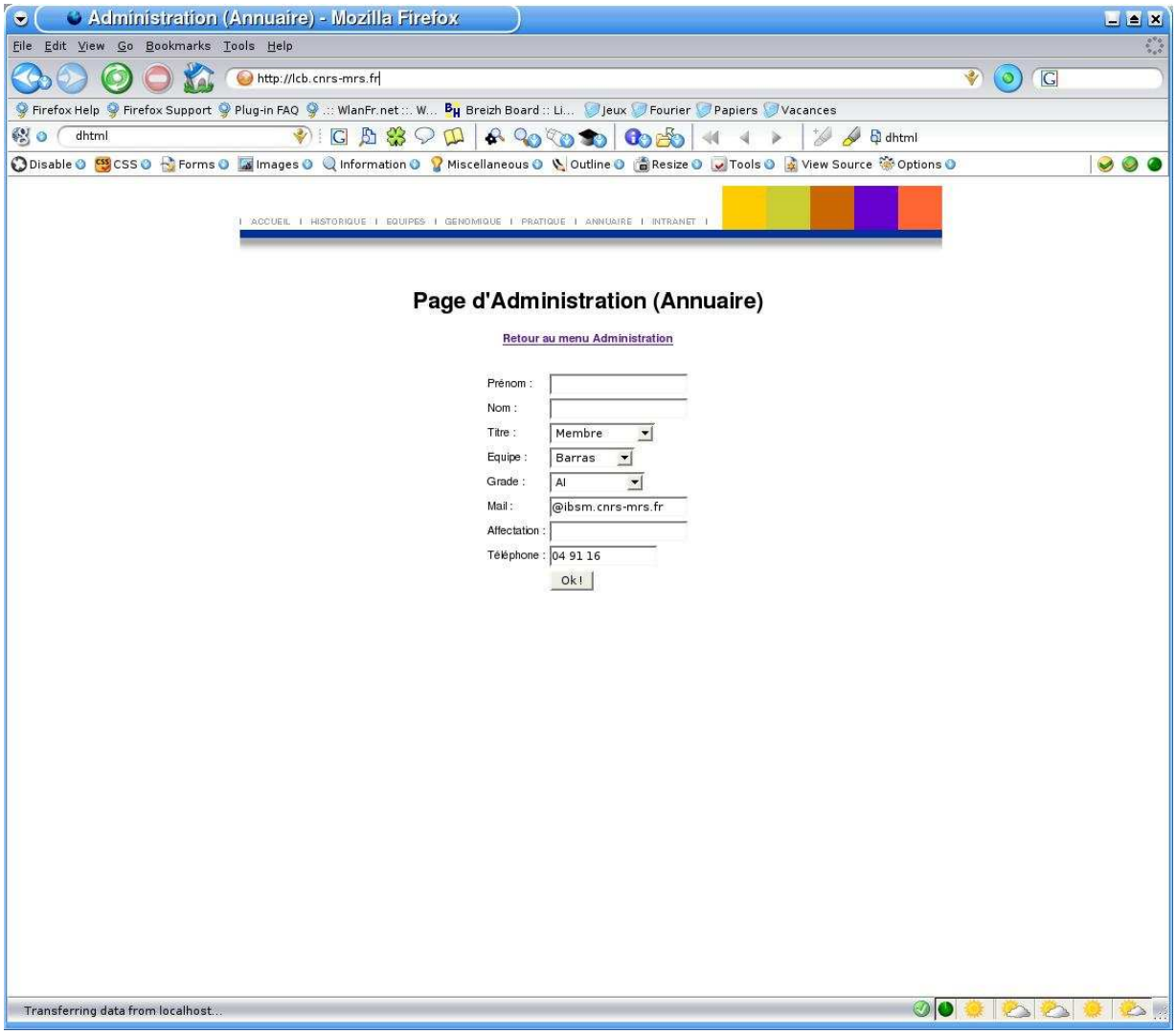


FIG. 7.12 – Ecran de saisie du personnel

```
}
if (!empty($_POST{"Action"})
{
    // Recherche si l'individu n'est pas déjà dans la base
    $resultat=ExecRequete("SELECT * FROM LCB.ANNUAIRE WHERE PRENOM=\"".
                          $_POST{"Prenom"}."\" AND NOM=\"".$_POST{"Nom"}.
                          "\"", $connexion);
    $elt=mysql_fetch_object($resultat);
    if (!empty($elt->ID))
    {
        // L'individu existe : possibilité de modifier les données le
        // concernant ou de le supprimer
        print "<TABLE>\n";
        print "<TR>\n";
        print "<TD><A href=\"annuaire.php?id=".$elt->ID."&amp;modif=1\">
               <IMG src=\"images/modif.png\" alt=\"Modif\" border=\"0\" />
               </A></TD><TD><A href=\"annuaire.php?id=".$elt->ID.
               "&amp;modif=2\"><IMG src=\"images/supp.png\" alt=\"Supp\"
               border=\"0\" /></A></TD><TD>".$_POST{"Prenom"}." ".
               $_POST{"Nom"}." (équipe ".$elt->EQUIPE.")</TD>\n";
        print "</TR>\n";
        print "</TABLE>\n";
        print "</DIV>\n";
        print "</BODY>\n";
        print "</HTML>\n";
        exit(0);
    }
    // Insertion des données dans la base
    $ident=determine_ID("LCB.ANNUAIRE", $connexion);
    $requete="INSERT INTO LCB.ANNUAIRE (ID, PRENOM, NOM, TITRE, GRADE,
              MAIL, AFFECTATION, EQUIPE, TEL) VALUES($ident, \"".
              $_POST{"Prenom"}."\", \"".$_POST{"Nom"}."\", \"".
              $_POST{"Titre"}."\", \"".$_POST{"Grade"}."\", \"".
              $_POST{"Mail"}."\", \"".$_POST{"Affectation"}.
              "\", \"".$_POST{"Equipe"}."\", \"".$_POST{"Tel"}."\")";
    ExecRequete($requete, $connexion);
}
if (!empty($_GET{"id"}))
{
    if ($_GET{"modif"}==1)
    {
        // Modification des données de l'individu
        $resultat=ExecRequete("SELECT * FROM LCB.ANNUAIRE WHERE ID=".
                              $_GET{"id"}, $connexion);
```

```
        $elts_a_modifier=mysql_fetch_object($resultat);
        $modification=1;
    }
    else
    {
        // Suppression de l'individu
        $resultat=ExecRequete("DELETE FROM LCB.ANNUAIRE WHERE
                               ID=".$_GET{"id"}, $connexion);
    }
}
print "<FORM method=\"POST\" action=\"annuaire.php\">\n";
print "<TABLE>\n";
print "<TR>\n";
if (!empty($modification))
    print "<TD>Prénom :</TD><TD><INPUT type=\"text\" name=\"Prenom\"
           size=\"20\" value=\"".$elts_a_modifier->PRENOM."\"/></TD>\n";
else
    print "<TD>Prénom :</TD><TD><INPUT type=\"text\" name=\"Prenom\"
           size=\"20\" /></TD>\n";
print "</TR>\n";
print "<TR>\n";
if (!empty($modification))
    print "<TD>Nom :</TD><TD><INPUT type=\"text\" name=\"Nom\"
           size=\"20\" value=\"".$elts_a_modifier->NOM."\"/></TD>\n";
else
    print "<TD>Nom :</TD><TD><INPUT type=\"text\" name=\"Nom\"
           size=\"20\" /></TD>\n";
print "</TR>\n";
print "<TR>\n";
print "<TD>Titre :</TD><TD><SELECT name=\"Titre\" size=\"1\">\n";
if (!empty($modification) && ($elts_a_modifier->TITRE != "Membre"))
{
    print "<OPTION>Membre</OPTION>\n";
    print "<OPTION selected>Responsable</OPTION>\n";
}
else
{
    print "<OPTION selected>Membre</OPTION>\n";
    print "<OPTION>Responsable</OPTION>\n";
}
print "</SELECT></TD>\n";
print "</TR>\n";
print "<TR>\n";
print "<TD>Equipe :</TD><TD><SELECT name=\"Equipe\" size=\"1\">\n";
```

```
$resultat=ExecRequete("SELECT NOM FROM LCB.EQUIPES", $connexion);
$passage=0;
while ($elt=mysql_fetch_object($resultat))
{
    if ((empty($passage) && !$modification) ||
        ($modification && $elts_a_modifier->EQUIPE==$elt->NOM))
    {
        $passage++;
        print "<OPTION selected>".$elt->NOM."</OPTION>\n";
    }
    else
        print "<OPTION>".$elt->NOM."</OPTION>\n";
}
print "</SELECT></TD>\n";
print "</TR>\n";
print "<TR>\n";
print "<TD>Grade :</TD><TD><SELECT name=\"Grade\" size=\"1\">\n";
$resultat=ExecRequete("SELECT ID FROM LCB.GRADES", $connexion);
$passage=0;
while ($elt=mysql_fetch_object($resultat))
{
    if ((empty($passage) && !$modification) ||
        ($modification && $elts_a_modifier->GRADE==$elt->ID))
    {
        $passage++;
        print "<OPTION selected>".$elt->ID."</OPTION>\n";
    }
    else
        print "<OPTION>".$elt->ID."</OPTION>\n";
}
print "</SELECT></TD>\n";
print "</TR>\n";
print "<TR>\n";
if (!empty($modification))
    print "<TD>Mail :</TD><TD><INPUT type=\"text\" name=\"Mail\"
           size=\"20\" value=\"".$elts_a_modifier->MAIL."\" /></TD>\n";
else
    print "<TD>Mail :</TD><TD><INPUT type=\"text\" name=\"Mail\"
           size=\"20\" value=\"@ibsm.cnrs-mrs.fr\" /></TD>\n";
print "</TR>\n";
print "<TR>\n";
if (!empty($modification))
    print "<TD>Affectation :</TD><TD><INPUT type=\"text\"
           name=\"Affectation\" size=\"20\" value=\"".
```

```
                    $elts_a_modifier->AFFECTATION."\"/></TD>\n";
                else
                    print "<TD>Affectation :</TD><TD><INPUT type=\"text\"
                           name=\"Affectation\" size=\"20\" /></TD>\n";
                print "</TR>\n";
                print "<TR>\n";
                if (!empty($modification))
                    print "<TD>Téléphone :</TD><TD><INPUT type=\"text\" name=\"Tel\"
                           size=\"15\" value=\"".$elts_a_modifier->TEL."\"/></TD>\n";
                else
                    print "<TD>Téléphone :</TD><TD><INPUT type=\"text\" name=\"Tel\"
                           size=\"15\" value=\"04 91 16 \"/></TD>\n";
                print "</TR>\n";
                print "<TR>\n";
                if (!empty($modification))
                {
                    print "<INPUT type=\"hidden\" name=\"ID\" value=\"".
                           $elts_a_modifier->ID."\" />\n";
                    print "<TD></TD><TD><INPUT type=\"submit\" name=\"Modification\"
                           value=\"Modifier !\" /></TD>\n";
                }
                else
                    print "<TD></TD><TD><INPUT type=\"submit\" name=\"Action\"
                           value=\"Ok !\" /></TD>\n";
                print "</TR>\n";
                print "</TABLE>\n";
                print "</FORM>\n";
            ?>
        </DIV>
    </BODY>

</HTML>
```

7.3 Les publications

Enfin, la gestion des publications s'effectue de la même manière que la gestion des actualités. Seules les données sauvegardées et le schéma de la table changent.

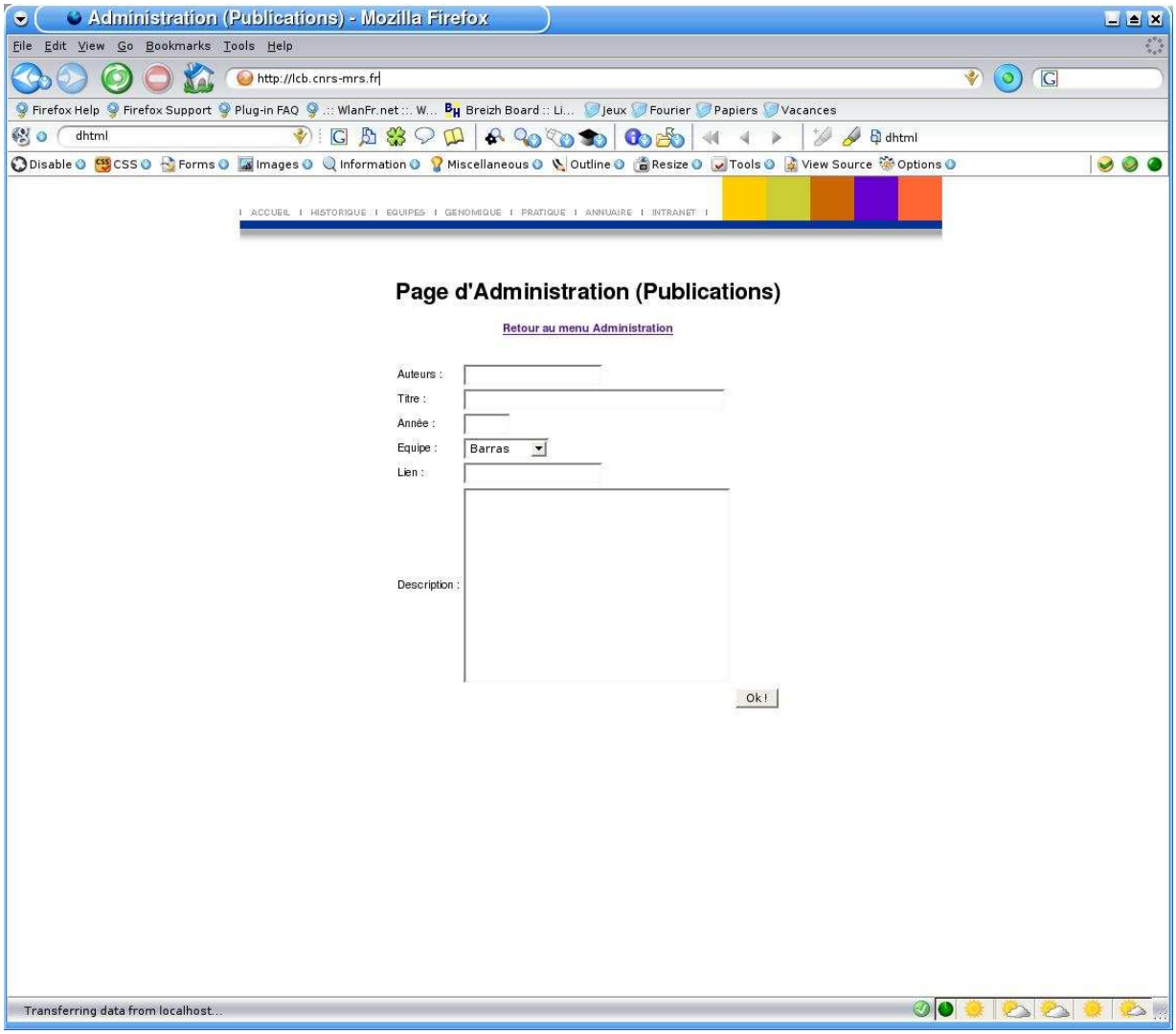


FIG. 7.13 – Ecran de saisie des publications

Conclusion

NOUS avons pu voir rapidement les différents langages que l'on pouvait mettre en oeuvre pour créer des sites dynamiques. Mais ceci peut être réalisé à plusieurs niveaux :

- De l'utilisation simple de PHP/MySQL enfoui dans du code HTML,
- Jusqu'à l'utilisation de feuilles de style XSLT dans du code PHP/MySQL, nécessitant la maîtrise de XML, XSLT, HTML, PHP, UML, SQL, et éventuellement du C ou du Perl pour faire des appels vers des programmes externes.

On peut donc voir que ces développements peuvent se révéler relativement complexes pour peu que l'on désire produire un logiciel évolutif s'appuyant sur des bases solides. Pour celà il n'y a pas de miracles, il faut mettre les mains dans le cambouis[17] et tester ...

[17] Les documentations de tous les logiciels installés se trouvent dans le répertoire /usr/share/doc.

Références

FPDF

`http://www.fpdf.org`
`http://www.phpteam.net`

HTML

HTML et XHTML – La référence, C. Musciano et B. Kennedy, 2001, 4ème édition, éd. O'Reilly.
`http://www.w3.org`

JPGraph

`http://www.aditus.nu/jpgraph`
`http://www.phpteam.net`

MySQL

Pratique de MySQL et PHP, P. Rigaux, 2003, 2ème édition, éd. O'Reilly.
Découvrez MySQL 5 et les procédures stockées, G. Lejeune, 2004, n°61, Linux Magazine.

PHP

Pratique de MySQL et PHP, P. Rigaux, 2003, 2ème édition, éd. O'Reilly.
PHP5 : Tout pour débuter et progresser avec PHP, Collectif, 2004, Hors Série n°20, Linux Magazine.
Programmation en Perl, L. Wall, T. Christiansen, et R. L. Schwartz, 1996, 2ème édition, éd. O'Reilly.
`http://www.php.net`
`http://www.nexen.net`

XML/XSLT/SAX/DOM

HTML et XHTML – La référence, C. Musciano et B. Kennedy, 2001, 4ème édition, éd. O'Reilly.
Pratique de MySQL et PHP, P. Rigaux, 2003, 2ème édition, éd. O'Reilly.
PHP5 : Tout pour débuter et progresser avec PHP, Collectif, 2004, Hors Série n°20, Linux Magazine.
Cours XML, A. Mkadmi, 2004, `http://h2ptm.hymedia.univ-paris8.fr/mkadmi/coursXML`.
`http://www.w3.org`
`http://www.w3schools.com`

Annexe A
Exercices

PHP

Les Bases du langage

1. Ecrire un petit script dans lequel on utilisera quatre variables a, b, c, et d. a et b contiennent des valeurs quelconques. On veut que c fasse référence à la valeur de a et d à la valeur de b. Résoudre ce problème de deux manières (affectation directe, et variables de variables). Quel problème pourrait-on rencontrer en utilisant une affectation directe ?
2. Quelle structure utiliser pour stocker les données de la table suivante :

Espece	**Gene**	**Domaine**	**Fonction**
Bacillus subtilis	yurY	1	A
Bacillus subtilis	yurY	2	B
Escerichia coli	dnaA	1	C
Yersinia Pestis	YP003	1	B
Yersinia Pestis	YP003	2	D

Structures de contrôle

1. Ecrire un script permettant de parcourir et d'afficher toutes les valeurs du tableau de l'exercice "2 – Les Bases du langage".
2. Ecrire de trois manières différentes un script affichant les nombres pairs de 0 à 10 compris, en utilisant :
   – une boucle `for` "normale",
   – une boucle `for` et un `continue`,
   – une boucle `while`.
3. Soit un tableau `$tab` de 100 éléments (`$tab[0]=1, ...$tab[99]=100`).

1	2	.....	100

   Lors de la lecture, on considère que l'élément suivant le dernier élément (`$tab[99]`) est le début du tableau. Donc, si l'on veut lire 10 valeurs à partir de la position 97 on provoquera l'affichage de : 98-99-100-1-2-3-4-5-6-7. Le nombre de valeurs à afficher sera donné dans la variable `$nb_val` et la position de départ sera donnée par `$start`. Ecrire un script permettant cet affichage.

Les fonctions

1. Définir une fonction qui étant donné trois entiers a, b, et c renvoie :
   – $a + b$ si $c = 0$,
   – $a - b$ si $c = 1$,
   – $a * b$ si $c = 2$,
   – a/b si $c = 3$ (erreur si $b = 0$).
   Si le paramètre c n'est pas spécifié, la fonction renvoie $a * b$.
2. Créer un script PHP qui recoit 3 paramètres (id, nom, et lien) en méthode GET. Ces paramètres seront récupérés par une fonction et transmis sous forme de table de hachage.
3. Créer un script qui récupère un paramètre donné "`Param`" en méthode `POST`. Une fonction devra tester si la chaîne contenue dans cette variable (passée en paramètre de la fonction) est de la forme (il s'agit du second paramètre de la fonction) :
   (a) Alternance de caractères majuscules et minuscules.
   (b) Identificateur (une lettre suivie d'une suite de lettres, chiffres ou "_").
   (c) Un mot composé de 3 chiffres, 4 lettres, et le signe "-" (ordonné).
   (d) Un mot composé de 3 chiffres, 4 lettres, et le signe "-" (ordre quelconque).

   Si la forme est correcte la fonction renverra Vrai et Faux sinon.
4. Créer une fonction qui prend en paramètre 2 chaînes de caractères "`fic`" et "`text`". Cette fonction va créer un fichier `fic.txt` contenant "`text`".
   Créer une seconde fonction qui nécessite un seul paramètre "`fic`ët qui va ouvrir le fichier `fic` (et non pas `fic.txt`), et l'afficher à l'écran sous forme de page HTML.
5. Créer une fonction qui ouvre un fichier dont le nom est passé en paramètre et qui le lit lettre à lettre et l'affiche à l'écran.
6. Créer une fonction qui ouvre un fichier dont le nom est passé en paramètre, qui lit les lettres par blocs de 20 et affiche à chaque itération le pourcentage du fichier lu.
7. Créer une fonction qui prend en paramètre une chaîne de caractères du type "`10 5 12 25 32`" et qui renvoie une chaîne contenant ces nombres multipliés par 3 (par exemple, ici : "`30 15 36 75 96`").

Notions de Programmation Orientée-Objet

Créer une classe `Annuaire` contenant les attributs : Nom, Prénom, Adresse_mail. Créer le constructeur de la classe ainsi qu'une méthode affichant le nom du serveur mail utilisé.

Les sessions & les cookies

1. On désire créer un site s'adaptant à la langue du navigateur utilisé. La détection de la langue ne devra être effectuée qu'une seule fois lors de l'ouverture de la première page. Quel problème pourra-t-on rencontrer avec un tel procédé ?
2. Créer un script qui récupère des paramètres "`Nom`" et "`Prenom`" passés en méthode `POST`. Créer ensuite deux cookies du même nom que les paramètres. Le premier expirera au bout de 6h, et le second au bout de 10min. Comme vous êtes distrait (si, si) vous effacerez le second cookie tout juste après l'avoir créé (de toute façon c'est un cookie et il vaut mieux utiliser les sessions ...).

PHP & MySQL

1. Créer un script permettant de se connecter à une base de données MySQL en fournissant un mot de passe. Le mot de passe valide sera stocké de manière cryptée dans la base elle-même.
2. Créer un script (sans aucune fonction !) qui se connecte à la base MySQL `MiddleGard` en tant que `Legolas` – mot de passe `Valinda` sur le serveur `Tolkien`, récupère tous les objets de la table `IsenGard` et les affiche.
3. Même question sur une base ORACLE.

Librairies PHP

La librairie FPDF

Créer un document PDF (à partir d'un script PHP bien sûr ...). Il s'agira d'un document en orientation "paysage" pour une page au format A4 et de titre "Création d'une page PDF".
La police utilisée sera du Times 12pts.
Le titre de la page sera affiché en rouge au centre de la page.
Afficher ensuite un petit texte (justifié) d'une dizaine de lignes.

La librairie JPGraph

Générer un graphe dont le titre "Test avec JPGraph" sera affiché en rouge. Il s'agit d'un graphe contenant deux diagrammes à barres issus de :

- `$bar1=array(25, 12, 10, 8, 4, 2, 1, 1, 2, 6, 8);`
- `$bar2=array(12, 28, 4, 30, 1, 10, 17, 22, 12, 15, 33);`

Le premier diagramme sera représenté en bleu et le second en vert.

Annexe B
TP

Dans cette partie sont présentés les sujets de TP de l'année 2005-2006.

PHP TP1

Gestion d'une DVDthèque (épisode I)

Pour gérer les différents DVD d'une collection, nous allons les répertorier. Les différents champs de référencement seront :

- Le titre
- Le réalisateur
- L'éditeur
- L'année de première parution
- Le nombre d'exemplaires

1. Manipulation de variables

1. Initialiser dans un script PHP une variable par champ de référencement.
2. Afficher les valeurs de ces variables à l'aide des commandes `echo`, `print`, et `printf`.

2. Gestion de formulaires HTML

1. Créer un formulaire permettant la saisie des valeurs de chacune des variables suivantes :
   - Le titre : Text
   - Le réalisateur : Text
   - L'éditeur : Text
   - L'année de première parution : Text
   - Le nombre d'exemplaires : Radio (entre 1 et 5, par défaut 1)

   On ajoutera également les champs suivants :
   - Un commentaire : TextArea
   - Le genre : Select (Policier, Comédie, Science Fiction, Dessin Animé)
2. Récupérer les informations transmises par ce formulaire dans un autre script PHP en utilisant tout d'abord la méthode GET puis la méthode POST.

3. Faire également afficher dans ce nouveau script les valeurs de ces variables.
4. Certains champs sont obligatoires, d'autres doivent être compris entre certaines valeurs : effectuer des tests sur les valeurs transmises et afficher éventuellement un message d'erreur. Les champs obligatoires sont : le titre, le réalisateur et l'année de parution qui doit de plus être comprise entre 1990 et 2005.
5. Dans le script de création du formulaire, modifier l'affichage des bouton radio du "nombre d'exemplaires" en utilisant une boucle `for` puis une boucle `while`.
6. Ajouter au script d'affichage un bouton permettant de retourner à la page précédente où le formulaire contiendra les anciennes valeurs.

PHP TP2 et TP3

Gestion d'une DVDthèque (épisode II)

1. Formulaire d'accueil (*accueil.php*)

1. Ecrire un formulaire comportant 2 boutons radio :
   – *Ajouter un DVD*
   – *Louer un DVD*
   Le formulaire comprendra également un bouton de soumission.
2. Le bouton de soumission va appeler un script *traitement.php* qui va traiter les choix effectués dans *accueil.php* :
   – *Ajouter un DVD* : Exécuter le script *ajoutDVD.php*
   – *Louer un DVD* : Exécuter le script *locationDVD.php*

2. Ajout d'un DVD (*ajoutDVD.php*)

1. Ecrire un formulaire permettant de saisir toutes les informations sur un DVD (penser au TP1 !).
2. Créer un champ "`hidden`" indiquant que l'on ajoute un DVD.
3. Le formulaire sera soumis au script *accueil.php*.

3. Sauvegarde des données (*accueil.php*)

1. Ouvrir une session.
2. Ajouter au formulaire un bouton radio "*Affichage*".
3. Vérifier que les variables issues du formulaire de *ajoutDVD.php* existent :
   – Si c'est le cas, récupérer les informations éventuellement déjà stockées dans la variable de session *tableDVD* (test d'existence). Ajouter les nouvelles données à la fin du tableau et le sauvegarder sous forme de variable de session.
   – Sinon, afficher le message "Vous ne venez pas de la page ajoutDVD.php".
4. Dans *traitement.php*, le choix de *Affichage* provoquera l'exécution de *affichage.php*.

4. Affichage des données (*affichage.php*)

1. Récupérer les variables de session.
2. Afficher le nombre d'éléments dans le tableau *tableDVD*.

3. Afficher le nombre total d'exemplaires.
4. Afficher tous les DVD (Titre et Réalisateur) sous forme d'un tableau HTML.
5. Afficher sous forme de tableau HTML tous les DVD dont le titre commence par "L" (Titre et Réalisateur).

5. Location d'un DVD (*locationDVD.php*)

1. Ouvrir une session.
2. Créer un formulaire qui affiche la liste des DVD précédés d'une case à cocher (*checkbox*).
3. Le formulaire sera re-soumis au même script qui dans ce cas affichera l'état courant de la liste de locations. (variables de session).

6. Déconnexion (*accueil.php*)

1. Ajouter un bouton *Déconnexion* dans le formulaire.
2. Dans *traitement.php*, le choix de *Déconnexion* entraînera une clôture de la session (suppression de toutes les variables de session).

7. Améliorations possibles

1. Ne permettre la location d'un DVD seulement s'il reste au moins un exemplaire disponible.
2. Ajouter dans le menu d'accueil un bouton *Retour de location*.

PHP TP4 et TP5

Gestion d'une DVDthèque (épisode III)

Nous allons développer une interface permettant de gérer la DVDthèque à partir de fichiers. Nous disposerons de deux fichiers :

- Le fichier `identification.txt` est un fichier texte contenant des informations séparées par le caractère deux points ":" : login, mot de passe, adresse mail, et identifiant utilisateur (0=utilisateur "normal", 1=administrateur). Voici un exemple de fichier :
  ```
  root:MoTdEpAsSe:root@root.com:1
  elessar:MiddleGard:elessar@middlegard.com:0
  ross:Dinosaures:ross@friends.com:0
  ```
- Le fichier `DVD.txt` est un fichier texte contenant les informations sur les DVD disponibles (ou non) à la location (informations séparées par un cractère *pipe* "|") : titre, réalisateur, éditeur, année de première parution, nombre d'exemplaires disponibles, nombre d'exemplaires total. Par exemple :
  ```
  Le Retour du Roi|Peter Jackson|New Line Cinema|2004|2|10
  Shrek 2|DreamWorks|DreamWorks Animation|2005|4|5
  ```

1. Identification de l'utilisateur

1. Créer un fichier `index.html` contenant un formulaire permettant à l'utilisateur de saisir son login et son mot de passe. Le bouton "submit" appelera un script `identification.php` en méthode POST.
2. Créer un script `identification.php` qui récupèrera les login et password de l'utilisateur. Ce script contiendra les fonctions :
   - `LectureID()` qui lira le fichier `identification.txt` et renverra les informations sous forme d'un tableau mixte dont la première partie est une table de hachage indicée par le login et les champs suivants sont définis dans un tableau. Par exemple, `$tab{root}[0]="MoTdEpAsSe"`, `$tab{root}[1]="root@root.com"`, et `$tab{root}[2]=1`.
   - `Identify()` qui prendra en paramètres la table renvoyée par `LectureID()` ainsi que le login et le password de l'utilisateur. Si l'utilisateur est référencé dans la table et qu'il a fournit le bon mot de passe, alors la fonction renvoie "1". Sinon, elle renvoie 0.
   - `Erreur()` qui affiche un message d'erreur en cas de tentative de connexion avec des paramètres erronés.

   Utiliser ces fonctions pour identifier un utilisateur. Si l'utilisateur est reconnu, afficher un message de bienvenue.

2. Affichage du menu

Suivant que l'utilisateur soit administrateur ou non, nous afficherons des menus différents (ces menus – ou formulaires – seront affichés à la place du message de bienvenue dans `identification.php`).

1. S'il s'agit d'un administrateur, le formulaire permettra d'ajouter une nouvelle référence au fichier `DVD.txt` (champs : titre, réalisateur, éditeur, année de première parution, nombre d'exemplaires total) et appelera le script `ajout.php`.
2. S'il s'agit d'un utilisateur, le formulaire affichera tous les titres de la DVDthèque avec en vis-à-vis deux cases à cocher permettant de rendre ou d'emprunter un DVD. L'affichage de ce formulaire se fera par l'intermédiaire de la fonction `LectureDVD()` qui lira le fichier `DVD.txt` et affichera les champs : titre, réalisateur, éditeur, année de première parution ainsi qu'une case à cocher "emprunter" (si le nombre d'exemplaires disponibles est supérieur à 0) et une case à cocher "rendre" (si le nombre d'exemplaires disponibles est inférieur au nombre d'exemplaires total). Ce formulaire appelera le script `update.php`.

3. Ajout d'une référence

Dans `ajout.php`, récupérer les paramètres issus du formulaire `identification.php` et créer une fonction `ajoutDVD()` permettant d'ouvrir le fichier `DVD.txt` et d'ajouter la nouvelle référence en fin de fichier.

4. Mise à jour

Dans `update.php`, récupérer les paramètres du formulaire `identification.php` et créer une fonction `modifDVD()` qui va lire le fichier `DVD.txt` et le modifier en fonction des paramètres donnés par l'utilisateur. Pour celà il faudra lire entièrement le fichier (et le stocker sous forme de tableau), modifier le tableau en fonction des paramètres de l'utilisateur, puis réécrire le fichier `DVD.txt` en entier.

5. Bonus

1. Lors de l'identification d'un utilisateur, enregistrer son login et son password sous forme de variables de sessions et à l'ouverture de chaque script vérifier que l'utilisateur est correctement enregistré (sinon afficher un message d'erreur).
2. Lorsqu'un administrateur enregistre une nouvelle référence, envoyer un email à tous les utilisteurs indiquant qu'une nouvelle référence est disponible (et bien sûr le nom du DVD ...). Attention : pour que la fonction `mail()` soit disponible, il faut que PHP ait accès au service sendmail sur le serveur. En cas d'utilisation d'un autre programme de mail (type qmail ou postfix) il faut utiliser d'autres API.

PHP TP6

Gestion d'une DVDthèque (épisode IV)

1. Reprendre le TP précédent (épisode III) et stocker les données non plus dans des fichiers mais dans des tables d'une base de données MySQL.
   La base contiendra deux tables :
   - Une table `IDENTIFICATION` :
     - `Login VARCHAR(10) NOT NULL`
     - `Password VARCHAR(10) NOT NULL`
     - `Email VARCHAR(30) NOT NULL`
     - `Status INT NOT NULL`
     - `PRIMARY KEY (Login)`

     Note : Pour le mot de passe, penser à utiliser la fonction SQL `crypt()`.
   - Une table `DVD` :
     - `Titre VARCHAR(100) NOT NULL`
     - `Realisateur VARCHAR(30) NOT NULL`
     - `Editeur VARCHAR(30)`
     - `Parution INT NOT NULL`
     - `Dispo INT NOT NULL`
     - `Total INT NOT NULL`
     - `PRIMARY KEY (Titre, Realisateur)`
2. Dans le script `identification.php` (affichage du menu), s'il s'agit d'un utilisateur "normal", ajouter un formulaire permettant de faire une recherche de DVD (mêmes champs que la table `DVD` excepté `Dispo` et `Total`). Ce formulaire appellera un script `recherche.php` qui affichera le résultat de la recherche sous forme de tableau.

PHP PROJET

Gestion supervisée des pages d'un site Internet

Les pages d'un site internet évoluent. Dans le cas des sites de structures contenant de nombreuses personnes (exemple : `http://www.lcb.cnrs-mrs.fr`), il peut être intéressant pour l'administrateur de déléguer la mise à jour des pages aux usagers (de manière simplifiée) tout en contrôlant ces mises à jour.

1. Format MySQL imposé

La base de données s'appellera `PROJET_ADMIN` et les tables suivantes vous sont imposées :

1.1. La table EQUIPES

EQUIPES	
NOM	VARCHAR(20)
INTITULE	VARCHAR(200)

- Le champs `NOM` désigne le nom d'une équipe. Il s'agit du nom de la personne responsable du groupe (voir `ANNUAIRE`).
- Le champs `INTITULE` indique l'activité de l'équipe (Recherche sur les bactéries, ...).

1.2. La table ANNUAIRE

ANNUAIRE	
ID	INTEGER
PRENOM	VARCHAR(50)
NOM	VARCHAR(50)
PWD	VARCHAR(50)
TITRE	VARCHAR(30)
EQUIPE	VARCHAR(20)

- Le champs `PWD` désigne le mot de passe utilisé par l'utilisateur pour se connecter en utilisant le login `NOM_PRENOM`. Ce mot de passe sera crypté lors de l'insertion dans la base grâce à la commande `crypt()` de PHP ou grâce à la commande `PASSWORD` de MySQL (à vous de choisir le mécanisme que vous préférez).
- Le champs `TITRE` contient la chaîne *Responsable* ou *Membre*.

1.3. La table PUBLIS

Publis	
ID	INTEGER
AUTEURS	VARCHAR(100)
TITRE	VARCHAR(100)
EQUIPE	VARCHAR(20)
LIEN	VARCHAR(50)
ANNEE	INT
DESCRIPTION	VARCHAR(200)

– Le champs `AUTEURS` contient les différents auteurs ayant participé à la rédaction de l'article.
– Le champs `EQUIPE` indique à quelle équipe appartient la publication.
– Le champs `LIEN` contient éventuellement une URL.

2. Architecture du site

Il existe 4 équipes différentes : A, B, C et D. Chacune de ces équipes dispose de 3 pages : introduction, équipe, thématiques et publications. Par exemple, pour l'équipe A, ces pages sont hébergées dans : `equipe/A/intro.php`, `equipe/A/equipe.php`, `equipe/A/thematiques.php` et `equipe/A/publications.php`.
La page `index.html` permet de sélectionner une équipe (arrivée sur la page d'introduction), puis, chaque page de l'équipe contiendra un menu permettant de se déplacer entre les différentes pages.

3. Affichage des membres d'une équipe (sur 1 Pt)

Les pages `equipe/.../equipe.php` permettent d'afficher les membres de l'équipe sélectionnée. On affichera tout d'abord les nom et prénom du Responsable puis ceux des membres de l'équipe classés par ordre alphabétique sur le nom et sur le prénom.

4. Affichage des publications d'une équipe (sur 1 Pt)

Les pages `equipe/.../publications.php` permettent d'afficher les publications de l'équipe sélectionnée. Les publications seront affichées par année décroissante.

5. Configuration automatique (sur 2 Pts – éliminatoire)

La gestion des pages se fera par l'intermédiaire de la page `admin/gestion.php`. Lors de l'ouverture de cette page, si la base LCB n'existe pas, elle sera créée (si le mot de passe root a été changé – par défaut "" –, demander le mot de passe puis effectuer les modifications).

6. Connexion (sur 1 Pt)

La page `admin/gestion.php` contiendra un formulaire (login et mot de passe) permettant d'accéder aux fonctions de modification des pages. Seul un "Responsable" peut modifier les pages d'un site. En cas d'authentification réussie, on chargera la page `admin/menu.php` permettant de sélectionner la page à modifier (introduction, thématiques ou publications).

7. Gestion des Publications (sur 1 Pt)

Affichage des publications de l'équipe sélectionnée avec possibilité de modifier ou d'effacer une publications. On proposera également un formulaire permettant d'insérer une nouvelle publication dans la base.

8. Gestion des pages Introduction et Thématiques (sur 4 Pts)

Déterminer la structure de la table contenant le code de ces pages pour chaque équipe. Les pages permettant les modifications (`admin/modif_intro.php` et `admin/modif_theme.php`) auront la structure indiquée en figure 7.14. Elles seront composées de deux frames : à gauche un aperçu et à droite une frame permettant de "coder" la page :

- Une fenêtre texte permettra d'écrire le texte.
- Un bouton d'aide fera apparaître une popup avec un texte d'aide.
- Des boutons de style (gras, italique, ...) permettront de changer le style (Fonction Javascript provoquant l'affichage d'une balise indiquant le style retenu qui portera sur l'ensemble des caractères compris entre les caractères spéciaux "`|`". Par exemple,
  `<GRAS>|Ce texte est en gras|, et là non !`.
- Un bouton "Image" permettra l'insertion d'une image. Trouvez un mécanisme permettant à l'utilisateur de parcourir ses fichiers en mode graphique pour insérer une image. Ce mécanisme peut avoir lieu lors de l'appui sur le bouton image ou lors de la validation de la page.
- Un bouton "Rafraîchir l'aperçu" permettant de mettre à jour la frame de gauche.
- Un bouton de validation de la page : un mail est envoyé à l'administrateur pour lui indiquer le nom de l'équipe, le nom de la page modifiée et un lien vers celle ci. La page modifiée n'est pas directement publiée sur le site – ou intégrée à la base de données. Elle est convertie en code HTML, puis elle est enregistrée dans un fichier temporaire à identifiant unique. Ce fichier sera détruit lors de la validation ou du rejet de la page par l'administrateur.

Lors des mises à jour de l'aperçu, on affiche dans la frame de gauche le code de la frame de droite traduit en code HTML.

9. Contrôle Administrateur (sur 3 Pts)

L'administrateur pourra se connecter sur sa page de contrôle (`admin/root/controle.php`) protégée par un fichier `.htacces` (login : "root", mot de passe : "empty"). Sur cette page

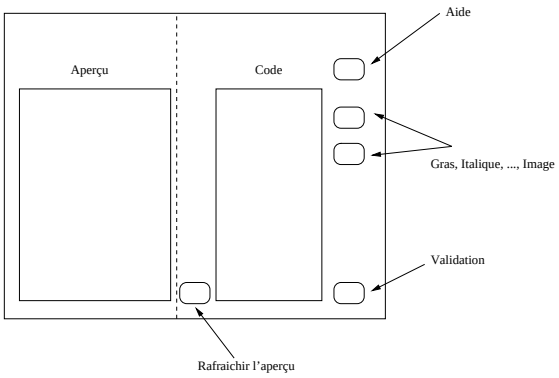


FIG. 7.14 – Ecran de modification des pages

apparaîtront les différentes modifications effectuées sur les pages et classées par ordre de date (les modifications les plus récentes en premier). Par exemple :

Date	Equipe	Page	Accepter	Rejeter
23/10/2005	Barras	introduction.php		

L'administrateur pourra alors accepter ou rejeter la modification (pour les pages introduction et thématiques) après les avoir visualisées (un lien permettra d'y accéder). Si la modification est acceptée elle sera alors publiée sur le site.
Un historique des modifications sera conservé et un rapport pourra être affiché sous forme PDF contenant un diagramme et un tableau des modifications par équipe.

10. Protection des pages de la partie administrateur (sur 1 Pt)

Protéger toutes les pages de la partie administrateur en utilisant des variables de session pour stocker les login et mot de passe.

11. Validitée du code HTML (éliminatoire)

Le code HTML devra être valide (petit bouton vert dans *HTML Validator* de FireFox).

12. Dernières consignes

Les contrôles des formulaires seront effectués en JavaScript (penser à créer une librairie).
Vous devrez rendre votre code par mail sous forme d'une archive `tar` compressée avec *gzip* (donc un fichier `.tar.gz`).
Le rapport sera noté sur 4 Pts et vous devrez y détailler vos choix de structure de base de donnée, et de programmation. Typiquement, un rapport devra contenir : une introduction, un schéma et des explications sur la structure de la base, les solutions de programmation retenues,

un manuel utilisateur, les améliorations apportées et les améliorations possibles et, enfin, une conclusion.
L'architecture du code (séparation en modules ou librairies et en fonctions), l'indentation et la présence de commentaires pertinents seront notés sur 2 Pts.

Annexe C
Annales TP

Dans cette partie est présenté le sujet de projet de l'année 2004-2005.

PHP 31/03/2005

Projet : Gestion d'un Club de Tennis

Le T. C. PHP, club de tennis bien connu, décide de se doter d'un outil de gestion des adhérents et des réservations de ses différents cours. Ces deux parties du projet sont totalement indépendantes et seront liées par la suite au sein d'une même interface.
Note : Vos pages devront respecter une DTD stricte (préférer une feuille de style externe commentée) et être validées (`http://validator.w3.org/`). De plus, l'emploi de code JavaScript est fortement recommandé.

1. Gestion des Adhérents

Lors de leur inscription, les adhérents de ce club remplissent une fiche contenant les informations suivantes ;

- Nom,
- Prénom,
- Sexe (M/F),
- Année de Naissance,
- Classement (NC, 30/5, 30/4, 30/3, 30/2, 30/1, 30, 15/5, 15/4, 15/3, 15/2, ou 15/1)
- Disponibilité :
  - Lundi : 8h-9h (O/N), ..., 20h-21h (O/N)
  - ...
  - Dimanche : 8h-9h (O/N), ..., 20h-21h (O/N)
- Numéro de téléphone,
- Numéro de membre (attribué par le Club)

Un administrateur identifié par un login et un mot de passe sera habilité à transférer ces informations sur le système informatisé. Les adhérents quant à eux pourront effectuer des recherches pour trouver un partenaire.

1.1. Menu général

Ce menu sera accessible à la fois à l'administrateur et aux adhérents.
L'administrateur doit s'identifier à l'aide d'un login et d'un mot de passe pour accéder à la partie "réservée" du site.
Un adhérent doit pouvoir effectuer une recherche dans la base des adhérents sur les critères suivants : Sexe, Age minimum, age maximum, Classement minimum, et Disponibilité.

1.2. Côté Administrateur

Les données relatives aux adhérents et à l'administration devront être stockées dans une base MySQL. Lorsque l'administrateur sera identifié, on affichera un formulaire permettant de saisir toutes les informations sur un adhérent. Si tous les champs sont renseignés, on insèrera les données dans la base. Sinon, on indiquera le nom des champs manquants et on proposera un retour vers le formulaire.

Bonus

- Utiliser des variables de session pour que l'administrateur n'ait pas à retaper son mot de passe à chaque nouvel adhérent ou erreur.
- En cas de champs manquant dans un formulaire, réafficher le formulaire pré-rempli.

1.3. Côté Adhérent

Afficher les résultats de la recherche dans la base de données (cf 1.1).

2. Gestion des Réservations

Le club dispose de 5 cours que les adhérents peuvent réserver à raison d'une seule réservation par adhérent et par jour pour un créneau d'une heure. Les réservations ne peuvent être effectuées que pour des jours de la semaine en cours en respectant les tranches : Lundi-Mardi-Mercredi-Jeudi et Vendredi-Samedi-Dimanche (par ex : on ne peut pas réserver du Jeudi pour le Vendredi, ni du Dimanche pour le Lundi, par contre on peut réserver du Lundi pour le Mercredi ...).
Développer l'interface permettant à un adhérent de réserver un cours (s'il n'est pas déjà réservé) pour lui et son partenaire (il faut 2 numéros de membre valides).

3. Interface générale

Créer une interface générale permettant d'accéder aux différents scripts.

4. Bonus

- Ajouter un script permettant à l'administrateur de générer un document PDF contenant les tableaux des réservations pour chaque cours.

– Ajouter un script permettant à l'administrateur d'afficher un diagramme dénotant l'occupation de chaque terrain dans la semaine.

5. Critères d'évaluation

Le projet sera noté sur 20 mais la note finale sera ramenée sur 5 : Partie 1 sur 6, Partie 2 sur 8, 2 points pour les parties "Bonus" et 4 points pour la lisibilité, l'organisation du code et le rapport qui devra contenir au moins :

– un descriptif des fonctionnalités implémentées
– l'architecture des scripts
– toute autre information que vous jugerez utile

Annexe D
Annales Examens

Dans cette partie sont présentés les sujets d'examens de l'année 2004-2005 accompagnés de leur correction.

Université d'Aix-Marseille I — IP6 : BD Internet
03/05/2005 — $1^{ère}$ session

Examen

Documents autorisés : Polycopié du cours, notes de cours
Durée : 2h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Ligue 1 de football (sur 10 pts)

Un journaliste de l'Equipe décide de faciliter ses analyses sur les résultats du championnat de football de ligue 1 en créant un script PHP. Les données qu'il souhaite utiliser sont représentées dans le journal sous la forme d'un tableau :

Equipe	**Pts**	**Journée**	**G**	**N**	**P**	**Bm**	**Be**	**Diff**
Marseille	70	24	23	1	0	61	10	51
Istres	65	24	21	2	1	45	25	20
Lens	45	23	12	9	2	23	14	9
Paris	3	23	0	3	20	1	52	-51
Monaco	52	24	15	7	2	37	36	1
Auxerre	22	24	7	1	16	28	32	-4
...	...	...	...	...	...	...	...	...

où G = nombre de matchs gagnés, N = nombre de matchs nuls, P = nombre de matchs perdus, Bm = buts marqués, Be = buts encaissés, et Diff = différence de buts.
Ces informations sont récupérées sur le site de l'AFP sous la forme d'un fichier texte où chaque champs est séparé par un caractère '|' (Marseille|70|24|...). C'est donc ce fichier texte qui devra être lu pour récupérer les informations.

1.1 Lire le fichier (sur 2 pts)

Ecrire une fonction *lecture* prenant comme paramètre le nom du fichier et qui renverra le tableau dans lequel on aura stocké les données (table de hachage de la forme : $tab{Equipe}{Pts}, $tab{Equipe}{Journée}, ...).

1.2 Classement général (sur 1 pt)

Ecrire une fonction *classement* prenant comme paramètre la table contenant les données (cf. 1.1) et qui affiche le nom des équipes par ordre de Pts décroissant.

1.3 Lanterne rouge (sur 1 pt)

Ecrire une fonction *lanterne_rouge* prenant comme paramètre la table contenant les données et qui renvoie le nom de la plus mauvaise équipe du championnat (celle qui à le moins de Pts).

1.4 Vérification (sur 2 pts)

Ecrire une fonction *verification* prenant comme paramètre la table contenant les données et qui va vérifier que les calculs sont corrects (que Diff = Bm - Be, que Pts = G*3 + N, et que Journée = G + N + P). Si les calculs sont faux, la table sera corrigée.

1.5 Recherche de lettres dans le nom des équipes (sur 2 pts)

Ecrire une fonction *enumeration* prenant comme paramètres la table contenant les données et un caractère. La fonction affichera la liste des équipes classées par ordre décroissant d'occurences du caractère dans le nom de chacune de ces équipes. Par exemple, si l'on recherche le caractère "e", on obtiendra :

```
Marseille -> 2
Auxerre -> 2
Istres -> 1
...
```

1.6 JPGraph ou FPDF (sur 2 pts)

Choisir et traiter une (et une seule des deux questions suivantes) :

- **JPGraph** : Ecrire un script qui lira un fichier grâce à la fonction *lecture* (cf. 1.1) et génèrera un diagramme camembert en 3D (jpgraph_pie3d) des buts marqués par chaque équipe.
- **FPDF** : Ecrire un script qui lira un fichier grâce à la fonction *lecture* et génèrera un document PDF contenant le tableau des résultats de ligue 1.

2. Le code inconnu (sur 5 pts)

Le code suivant a été récupéré sur internet. Il comporte de nombreuses erreurs : retrouvez les, expliquez les et dites ce que fait ce programme (attention, la répétition d'une même erreur sur plusieurs lignes ne compte qu'une seule fois).

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            <?PHP
9                <!-- Début du code PHP -->
10               $log=$_GET{"login"};
11               $pwd=$_POST{"pwd"};
12               print "\t\t\tVoici les paramètres récupérés dans l'URL (?) : <BR />\n
13                      \t\t\tLogin : $log, et Password : $pwd<BR />\n";
14           ?>
15           Et maintenant une table :<BR />
16           <?PHP
17               for ($i=0; $i<=10; $i++)
18                    table{"$i x 5"}=$ix5;
19               foreach (table as $elt)
20                    print "$elt<BR />\n";
21               print "<BR />Et enfin, une surprise!<BR />\n";
22               $fic=fopen("r", "file.txt");
23               if (empty($fic))
24               while (feof($fic))
25               {
26                    $ligne=fgets($fic, 1024);
27                    ereg($ligne, "/Debian/", $TabOcc);
28                    $cpt+=count($TabOcc)-1;
29               }
30               fclose($fic);
31      </BODY>
32 </HTML>
```

3. Base de données "Notes d'examen" (sur 5 pts)

On dispose d'un compte sur un serveur MySQL (Nom : "LicPro", Mot de passe : "20àl'exam", Serveur : "CMI") et on travaille sur la base "Examen" qui contient une table "RESULTATS" contenant les champs : Nom, Prenom, Exam_PHP, Projet_PHP, Final_PHP (seul le champs

Final_PHP est vide). Sachant que la note finale de l'examen sera calculée comme suit :
`Final_PHP=(3*Exam_PHP+Projet_PHP)/4`
Créez un script permettant de se connecter à la base et de remplir le champs Final_PHP.

Université d'Aix-Marseille I — IP6 : BD Internet
03/05/2005 — $1^{ère}$ session

Examen (corrigé)

Documents autorisés : Polycopié du cours, notes de cours
Durée : 2h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Ligue 1 de football (sur 10 pts)

1.1 Lire le fichier (sur 2 pts)

```
function lecture($fichier)
{
  $fic=fopen($fichier, "r");
  if (empty($fic))
  {
    print "Erreur lors de l'ouverture de $fichier<BR />\n";
    exit;
  }
  while (!feof($fic))
  {
    $ligne=fgets($fic, 1024);
    $tab_ligne=explode("|", $ligne);
    $tab{$tab_ligne[0]}{"Pts"}=$tab_ligne[1];
    $tab{$tab_ligne[0]}{"Journée"}=$tab_ligne[2];
    $tab{$tab_ligne[0]}{"G"}=$tab_ligne[3];
    $tab{$tab_ligne[0]}{"N"}=$tab_ligne[4];
    $tab{$tab_ligne[0]}{"P"}=$tab_ligne[5];
    $tab{$tab_ligne[0]}{"Bm"}=$tab_ligne[6];
    $tab{$tab_ligne[0]}{"Be"}=$tab_ligne[7];
    $tab{$tab_ligne[0]}{"Diff"}=$tab_ligne[8];
  }
  return($tab);
}
```

1.2 Classement général (sur 1 pt)

```
function classement($table)
```

```
{
  foreach ($table as $cle => $tab)
    $T{$cle}=$tab{"Pts"};
  arsort($T);
  foreach ($T as $cle => $val)
    print "$val => $cle<BR />\n";
}
```

– ou –

```
function classement($table)
{
  arsort($table);
  foreach ($table as $cle => $tab)
    print "$cle => $tab{"Pts"}<BR />\n";
}
```

1.3 Lanterne rouge (sur 1 pt)

```
function lanterne_rouge($table)
{
  $mini=-1;
  foreach ($table as $cle => $tab)
  {
    if ($tab{"Pts"} < $mini)
    {
      $mini=$tab{"Pts"};
      $lr=$cle;
    }
  }
  return($lr);
}
```

1.4 Vérification (sur 2 pts)

```
function verification($table)
{
  foreach ($table as $cle => $tab)
  {
    $diff=$tab{"Diff"};
    $Bm=$tab{"Bm"};
    $Be=$tab{"Be"};
    $Pts=$tab{"Pts"};
    $G=$tab{"G"};
    $P=$tab{"P"};
```

```
    $Jour=$tab{"Journée"};
    $table{$cle}{"Diff"}=$Bm-$Be;
    $table{$cle}{"Pts"}=3*$G+$N;
    $table{$cle}{"Journée"}=$G+$N+$P;
  }
  return($table);
}
```

1.5 Recherche de lettres dans le nom des équipes (sur 2 pts)

```
function enumeration($table, $car)
{
  foreach ($table as $nom => $tab)
  {
    ereg($car, $nom,  $TabOcc);
    $T{$nom}=count($TabOcc)-1;
  }
  arsort($T);
  foreach ($T as $cle => $val)
    print "$cle -> $val<BR />\n";
}
```

1.6 JPGraph ou FPDF (sur 2 pts)

Choisir et traiter une (et une seule des deux questions suivantes) :

- **JPGraph :**

```
<?PHP
  include("/usr/share/jpgraph/jpgraph.php");
  include("/usr/share/jpgraph/jpgraph_pie3d.php");

  $i=0;
  $table=lecture("fichier.txt");
  foreach ($table as $cle => $tab)
    $data[$i++]=$tab{"Bm"};
  $graph=new PieGraph(300, 200);
  $pie=new PiePlot($data);
  $pie->SetTheme("earth");
  $pie->Explode(array(0, 15, 15, 25, 15));
  $graph->Add($pie);
  $graph->Stroke();
?>
```

- **FPDF :**

```
<?PHP
  define(FPDF_FONTPATH, "/usr/share/fpdf/font/");
```

```
require("/usr/share/fpdf/fpdf.php");

$table=lecture("fichier.txt");
$pdf=new FPDF("P", "mm", "A4");
$pdf->AddPage();
$pdf->SetFont("Arial", "B", "16");
$pdf->Cell(50, 10, "Equipe", 1, 0, "C");
$pdf->Cell(50, 10, "Pts", 1, 0, "C");
$pdf->Cell(50, 10, "Journée", 1, 0, "C");
$pdf->Cell(50, 10, "G", 1, 0, "C");
$pdf->Cell(50, 10, "N", 1, 0, "C");
$pdf->Cell(50, 10, "P", 1, 0, "C");
$pdf->Cell(50, 10, "Bm", 1, 0, "C");
$pdf->Cell(50, 10, "Be", 1, 0, "C");
$pdf->Cell(50, 10, "Diff", 1, 0, "C");
$pdf->Ln();
$pdf->SetFont("Arial", "", "16");
foreach ($table as $cle => $tab)
{
  $pdf->Cell(50, 10, $cle, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"Pts"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"Journée"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"G"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"N"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"P"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"Bm"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"Be"}, 1, 0, "C");
  $pdf->Cell(50, 10, $tab{"Diff"}, 1, 0, "C");
  $pdf->Ln();
}
$pdf->OutPut();
?>
```

2. Le code inconnu (sur 5 pts)

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            <?PHP
9                <!-- Début du code PHP -->
                 // Les commentaires se font à l'aide de // (0.25 pts)
10               $log=$_GET{"login"};
11               $pwd=$_POST{"pwd"};
                 // Ici paramètre doit être $pwd=$_GET{"pwd"} (0.25 pts)
12               print "\t\t\tVoici les paramètres récupérés dans l'URL (?) : <BR />\n
13                      \t\t\tLogin : $log, et Password : $pwd<BR />\n";
14           ?>
15           Et maintenant une table :<BR />
16           <?PHP
17               for ($i=0; $i<=10; $i++)
18                    table{"$i x 5"}=$ix5;
                 // $table et $i*5 au lieu de $ix5 (0.25 + 0.5 pts)
19               foreach (table as $elt)
                 // foreach ($table as $cle => $elt) (1 pt)
                 // car il s'agit d'une table de hachage
20                    print "$elt<BR />\n";
21               print "<BR />Et enfin, une surprise!<BR />\n";
22               $fic=fopen("r", "file.txt");
                 // $fic=fopen("file.txt", "r"); (0.25 pts)
23               if (empty($fic))
                 // if (!empty($fic)) (0.25 pts)
24               while (feof($fic))
25               {
26                    $ligne=fgets($fic, 1024);
27                    ereg($ligne, "/Debian/", $TabOcc);
                      // ereg("Debian", $ligne, $TabOcc); (0.25 pts)
28                    $cpt+=count($TabOcc)-1;
29               }
30               fclose($fic);
               // Balise fermante ?> manquante (0.25 pts)
31      </BODY>
32 </HTML>
```

Ce programme récupère deux paramètres dans l'URL (`log` et `pwd`), affiche la table de multiplication par 5 de 0x5 à 10x5, et enfin compte le nombre d'occurences du mot `Debian` dans le fichier `file.txt` (0.5 + 0.5 + 0.5 pts).

3. Base de données "Notes d'examen" (sur 5 pts)

```
function Notes_Exam()
{
  $connexion=mysql_pconnect("CMI", "LicPro", "20àl'exam");
  if (empty($connexion))
  {
    print "Echec de la connexion ...<BR />\n";
    exit;
  }
  if (!mysql_select_db("Examen", $connexion))
  {
    print "Accès à la base impossible ...<BR \>\n";
    exit;
  }
  $resultat=mysql_query("SELECT Nom, Prenom, Exam_PHP, Projet_PHP FROM
                         EXAMEN.RESULTATS", $connexion);
  while ($elt=mysql_fetch_object($resultat))
  {
    $final=(3*$elt->Exam_PHP+Projet_PHP)/4;
    $requete="UPDATE EXAMEN.RESULTATS SET Final_PHP=\"".$final."\" WHERE
              Nom=\"".$elt->Nom."\" AND Prenom=\"".$elt->Prenom."\"";
    $res=mysql_query($requete, $connexion) or die(mysql_error());
  }
}
```

Université d'Aix-Marseille I — Outils de l'Internet
31/05/2005 — $1^{ère}$ session

Examen

Documents autorisés : Polycopié du cours, notes de cours
Durée : 3h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Conception de grilles de Mots-Fléchés (sur 6 pts)

Une société décide de produire en masse des grilles de mots-fléchés. Pour celà, elle va développer un logiciel en PHP permettant de générer ces grilles à partir d'une base de données fournie par une société tierce. Voici les informations dont vous disposez :

```
SERVEUR :          MYSQL_DREAMMACHINE
NOM DE LA BASE :   MF
LOGIN :            CR_Grille
PASSWORD :         CMI

NOM DE LA TABLE : MOTS_FLECHES
CHAMPS :
    - MOT (VARCHAR(30), PRIMARY KEY) : mot à découvrir
    - NBRE_LETTRES (INT)             : nombre de lettres du mot
    - DEF (VARCHAR(30))              : courte définition permettant de découvrir
                                       le mot
```

Par exemple,

MOT	NBRE_LETTRES	DEF
crues	5	grossières
ode	3	poésie lyrique
décapitées	10	étêtées
lie	3	fond de cuve
...	...	...

1.1 Fonction d'accès à la base (sur 0.25 pt)

Ecrire une fonction *connexion* permettant de se connecter à la base et renvoyant l'identifiant de connexion.

1.2 Importation d'un fichier texte (sur 0.75 pt)

Ecrire une fonction *import* prenant comme paramètre le nom d'un fichier texte (au format `MOT|NBRE_LETTRES|DEF`) et qui insère ces données dans la base. De plus, cette fonction devra afficher à l'écran un tableau contenant les données.

1.3 Tri des mots (sur 0.5 pt)

Ecrire une fonction *trier* qui affiche les mots de la base par ordre alphabétique croissant.

1.4 Recherche de lettres dans un mot (sur 1.5 pts)

Ecrire une fonction *enumeration* prenant comme paramètre une chaîne de caractères. La fonction affichera la liste des lettres qui composent ce mot classées par ordre décroissant d'occurences. Par exemple, pour le mot *décapitées* :

```
é -> 2
a -> 1
c -> 1
d -> 1
e -> 1
i -> 1
p -> 1
s -> 1
t -> 1
```

1.5 Intersection entre mots (sur 1.5 pts)

Ecrire une fonction *intersection* prenant comme paramètres deux chaînes de caractères et qui renvoie dans une table les lettres communes aux deux chaînes. Par exemple, avec *dé**c**apité**es*** et ***c**ru**es***, on obtiendrait le tableau :

```
$tableau[0]="c"
$tableau[1]="e"
$tableau[2]="s"
```

1.6 Création d'une grille en PDF (sur 1.5 pts)

Nous supposerons ici que nous disposons d'un tableau T et d'une table de hachage D contenant toutes les informations concernant une grille de mots fléchés (de 20 x 30 lettres) :

```
$T[i][j] => lettre de la case (i,j)
si $T[i][j] est vide, c'est qu'il s'agit d'une définition se trouvant dans la
table $D{"i,j"}
```

A partir de ces informations, générer un document PDF contenant une grille vide (avec les définitions mais sans les mots) et une grille de correction.

2. Le code inconnu (sur 5 pts)

Le code suivant a été récupéré sur internet. Il comporte de nombreuses erreurs : retrouvez-les, expliquez-les et dites ce que fait ce programme (attention, la répétition d'une même erreur sur plusieurs lignes ne compte qu'une seule fois).

```
1  <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD name="Entete">
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            <?PHP
9                /* Début du code PHP
10               function whatsthat($T)
11               {
12                   $f=file($T);
13                   if (empty($f))
14                   {
15                       $t=explode(" ", $f, $T);
16                       foreach ($t as $c)
17                           $tab{$c}+=1;
18                   }
19                   return($tab);
20               }
21               $m=whatsthat("fichier.txt");
22               define(FPDF_FONTPATH, "/usr/share/fpdf/font/");
23               require("/usr/share/fpdf/fpdf.php");
24               foreach ($val as $m)
25               {
26                   $pdf->Cell(50, 10, $m, 1, 0, "C");
27                   $pdf->Ln();
28               }
29               $PDF->OutPut();
30      </BODY>
31 </HTML>
```

3. Un peu de XML (sur 3 pts)

Ecrire un script permettant de lire un fichier *"fic.txt"* ayant la structure suivante (la première ligne est donnée à titre indicatif, elle ne fait pas partie du fichier) :

```
Bactérie|Chromosome|Gène
Bacillus subtilis|A|yurY
Yersinia pestis|A|YP0012
Escherichia coli|A|malK
Yersinia pestis|B|YP0567
Yersinia pestis|A|YP0002
```

Note : Les chromosomes auront pour nom A, B, C, ou D.
Une fois le fichier lu, le stocker dans une structure appropriée, et l'afficher sous forme XML en l'ayant trié par ordre alphabétique sur le nom des bactéries, des chromosomes et des gènes. Voici un exemple de sortie attendue :

```
<BACTERIE>
  Yersinia pestis
  <CHROMOSOME>
    A
    <GENE>
      YP0002
    </GENE>
    <GENE>
      YP0012
    </GENE>
  </CHROMOSOME>
  <CHROMOSOME>
    B
    <GENE>
      YP0567
    </GENE>
  </CHROMOSOME>
</BACTERIE>
```

Enfin, donner la DTD de ce fichier (l'élément racine pourra être *EXAM*).

4. Un compteur de visiteurs (sur 2.5 pts)

Ecrire un script permettant de compter le nombre de passages d'un visiteur sur une page. Attention, l'utilisateur ne sera compté qu'une fois par jour !
Vous êtes libre d'utiliser les structures que vous désirez, base de données, ... à condition de l'expliquer ...

5. Expressions Régulières (sur 3.5 pts)

Un script reçoit des variables à travers une méthode POST :

- un numéro de téléphone `$_POST{"Tel"}` : *xx.xx.xx.xx.xx* où x représente un nombre
- une adresse mail `$_POST{"Mail"}` : *xxxxx@yyyyyy.zzz* où xxxxx est un mot (ensemble de lettres et de chiffres) pouvant être de la forme x1.x2, yyyyyy est un mot et zzz une extension pouvant contenir 2 ou 3 lettres.
- un nom `$_POST{"Nom"}` : *Nom Prénom* où Nom et Prénom sont deux mots (ensemble de lettres) dont la première lettre est en majuscule et les suivantes en minuscule. Ces deux mots sont séparés par un espace.
- une équation `$_POST{"Equation"}` : $a\ b = c$ où a est un entier, b est un entier précédé d'un opérateur (pouvant se répéter plusieurs fois) et c est un entier.

Ecrire un script PHP qui vérifie que les variables qui lui sont passées correspondent au format requis (et affiche un message d'erreur sinon).

Université d'Aix-Marseille I
31/05/2005

Outils de l'Internet
$1^{ère}$ session

Examen (corrigé)

Documents autorisés : Polycopié du cours, notes de cours
Durée : 3h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Conception de grilles de Mots-Fléchés (sur 6 pts)

1.1 Fonction d'accès à la base (sur 0.25 pt)

```
function connexion()
{
  $connexion=mysql_pconnect("MYSQL_DREAMMACHINE", "CR_Grille", "CMI");
  if (empty($connexion))
  {
    print "Echec de la connexion ...<BR />\n";
    exit;
  }
  if (!mysql_select_db("MF", $connexion))
  {
    print "Accès à la base impossible ...<BR \>\n";
    exit;
  }
  return($connexion);
}
```

1.2 Importation d'un fichier texte (sur 0.75 pt)

```
function import($fichier, $connexion)
{
  $f=fopen($fichier, "r");
  if (empty($f))
  {
    print "Impossible d'ouvrir le fichier $f ...<BR />\n";
    exit;
  }
```

```
  print "<TABLE>\n";
  print "<TR><TH>MOT</TH><TH>NBRE_LETTRES</TH><TH>DEF</TH></TR>\n";
  while (!feof($f))
  {
    $ligne=fgets($f, 1024);
    chop($ligne);
    $tab=explode("|", $ligne);
    $requete="INSERT INTO MF.MOTS_FLECHES (MOT, NBRE_LETTRES, DEF) VALUES
              ('".$tab[0]."', '".$tab[1]."', '". $tab[2]."');";
    $resultat=mysql_query($requete, $connexion);
    if (empty($resultat))
      die("Erreur d'exécution de la requête ...<BR />\n");
    print "<TR><TD>$tab[0]</TD><TD>$tab[1]</TD><TD>$tab[2]</TD></TR>\n";
  }
  print "</TABLE>\n";
}
```

1.3 Tri des mots (sur 0.5 pt)

```
function trier($connexion)
{
  $requete="SELECT MOT FROM MF.MOTS_FLECHES;";
  $resultat=mysql_query($requete, $connexion);
  while ($elt=mysql_fetch_object($resultat))
    $tab[$i++]=$elt->MOT;
  sort($tab);
  for ($j=0; $j<$i; $j++)
    print $tab[$j]."<BR />\n";
}
```

1.4 Recherche de lettres dans un mot (sur 1.5 pts)

```
function enumeration($chaine)
{
  $lng=count($chaine);
  for ($i=0; $i<$lng; $i++)
    $tab{$chaine[$i]}++;
  rsort($tab);
  foreach ($tab as $cle => $val)
    print "$val -> $cle<BR />\n";
}
```

1.5 Intersection entre mots (sur 1.5 pts)

```
function intersection($ch1, $ch2)
{
  $lg1=count($ch1);
  $lg2=count($ch2);
  for ($i=0; $i<$lg1; $i++)
  {
    for ($j=0; $j<$lg2; $j++)
      if ($ch1[$i] == $ch2[$j])
        $tab[$k++]=$ch1[$i];
  }
  return($tab);
}
```

1.6 Création d'une grille en PDF (sur 1.5 pts)

```
<?PHP
  define(FPDF_FONTPATH, "/usr/share/fpdf/font");
  require("/usr/share/fpdf/fpdf.php");
  $pdf=newFPDF("L", "mm", "A4");
  $pdf->AddPage();
  $pdf->SetFont("Arial", "B", "16");
  // Grille vide
  for ($i=0; $i<20; $i++)
  {
    for ($j=0; $j<30; $j++)
      if (empty($T[$i][$j]))
        $pdf->Cell(50, 10, $D{\"".$i.",".$j."\"}, 1, 1, "C");
      else
        $pdf->Cell(50, 10, "", 1, 1, "C");
    $pdf->Ln();
  }
  // Grille de correction
   for ($i=0; $i<20; $i++)
  {
    for ($j=0; $j<30; $j++)
      if (empty($T[$i][$j]))
        $pdf->Cell(50, 10, $D{\"".$i.",".$j."\"}, 1, 1, "C");
      else
        $pdf->Cell(50, 10, $T[$i][$j], 1, 1, "C");
    $pdf->Ln();
  }
  $pdf->OutPut();
```

```
?>
```

2. Le code inconnu (sur 5 pts)

Le code suivant a été récupéré sur internet. Il comporte de nombreuses erreurs : retrouvez-les, expliquez-les et dites ce que fait ce programme (attention, la répétition d'une même erreur sur plusieurs lignes ne compte qu'une seule fois).

```
1  <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
   <!-- manque le ! devant DOCTYPE (0.25 pts) -->
3  <HTML>
4       <HEAD name="Entete">
        <!-- pas d'attribut name pour HEAD (0.25 pts) -->
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            <?PHP
9                /* Début du code PHP
                 // Le commentaire n'est pas fermé par */ (0.25 pts)
10               function whatsthat($T)
11               {
12                   $f=file($T);
13                   if (empty($f))
                     // c'est le test if (!empty($f)) (0.25 pts)
14                   {
                         // il faut parcourir tout le fichier
                         // les lignes 15, 16 et 17 sont dans une boucle
                         // foreach ($f as $k) (0.5 pts)
15                       $t=explode(" ", $f, $T);
                         // trop d'arguments pour explode :
                         // $t=explode(" ", $k); (0.5 pts)
16                       foreach ($t as $c)
17                           $tab{$c}+=1;
18                   }
19                   return($tab);
20               }
21               $m=whatsthat("fichier.txt");
22               define(FPDF_FONTPATH, "/usr/share/fpdf/font/");
23               require("/usr/share/fpdf/fpdf.php");
                 // pour générer un pdf il ne faut avoir que des instructions
                 // relatives à la construction du document PDF
                 // ces lignes devraient être en début de fichier (0.25 pts)
                 // il faut également créer l'objet $pdf :
```

```
                // $pdf=new FPDF ('P', 'mm', 'A4'); (0.5 pts)
                // $pdf->AddPage(); (0.25 pts)
24              foreach ($val as $m)
                // foreach ($m as $cle => $val) (1 pt)
25              {
26                  $pdf->Cell(50, 10, $m, 1, 0, "C");
                    // $pdf->Cell(50, 10, $cle." => ".$val, 1, 0, "C"); (0.25 pts)
27                  $pdf->Ln();
28              }
29              $PDF->OutPut();
                // L'objet $PDF n'existe pas : c'est $pdf
                // $pdf->Output(); (0.25 pts)
                // manque la balise fermante du PHP ?> (0.25 pts)
30      </BODY>
31 </HTML>
```

3. Un peu de XML (sur 3 pts)

```
<?PHP

  $fic=file("fic.txt");
  $t_fic=explode("\n", $fic);
  foreach ($t_fic as $ligne)
  {
    $tt_fic=explode("|", $ligne);
    $tab{$tt_fic[0]}{$tt_fic[1]}[$indice{$tt_fic[0]}{$tt_fic[1]}]=$tt_fic[2];
    $indice{$tt_fic[0]}{$tt_fic[1]}+=1;
  }

  print "<HTML>\n";
  print "  <BODY>\n";
  ksort($tab);
  foreach ($tab as $bact => $tab2)
  {
    print "<BACTERIE><BR />\n";
    print "\t$bact<BR />\n";
    ksort($tab2);
    foreach ($tab2 as $chromo => $tab3)
    {
      print "\t<CHROMOSOME><BR />\n";
      print "\t\t$chromo<BR />\n";
      sort($tab3);
      foreach ($tab3 as $val)
```

```
      {
        print "\t\t<GENE><BR />\n";
        print "\t\t\t$val<BR />\n";
        print "\t\t</GENE><BR />\n";
      }
      print "\t</CHROMOSOME><BR />\n";
    }
    print "</BACTERIE><BR />\n";
  }
  print "  </BODY>\n";
  print "</HTML>\n";

?>
```

Et la DTD :

```
<!ELEMENT EXAM (BACTERIE+)>

<!ELEMENT BACTERIE (CHROMOSOME+, GENE+)>
<!ELEMENT CHROMOSOME (A|B|C|D) #REQUIRED>
<!ELEMENT GENE (#PCDATA)>
```

4. Un compteur de visiteurs (sur 2.5 pts)

```
<?PHP
  setcookie("Passe", "1", time()+24*3600);
  if (empty($_COOKIE{"Passe"}))
  {
    $connexion=mysql_pconnect("MYSQL_DREAMMACHINE", "CR_Grille", "CMI");
    if (empty($connexion))
    {
      print "Echec de la connexion ...<BR />\n";
      exit;
    }
    if (!mysql_select_db("MF", $connexion))
    {
      print "Accès à la base impossible ...<BR \>\n";
      exit;
    }
    $requete="SELECT Nbre_Visiteurs FROM MF.VISITEURS";
    $resultat=mysql_query($requete, $connexion);
    $elt=mysql_fetch_object($resultat);
    $N=$elt->Nbre_Visiteurs + 1;
    $requete="INSERT INTO MF.VISITEURS (Nbre_Visiteurs) VALUES ($N);";
```

```
    $resultat=mysql_query($requete, $connexion);
  }
  /**********************
       Suite du code
  **********************/
?>
```

5. Expressions Régulières (sur 3.5 pts)

```
<?PHP
  // Tel => 0.5 pts
  if (ereg("[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}", $_POST{"Tel"}))
    print "Téléphone OK<BR \>\n";
  // Mail => 1.5 pts
  if (ereg("[A-Za-z0-9]+(\.[A-Za-z0-9]+){0,1}\.[A-Za-z0-9]+\.[A-Za-z0-9]{2,3}",
            $_POST{"Mail"}))
    print "Mail OK<BR \>\n";
  // Nom => 0.5 pts
  if (ereg("[A-Z][a-z]+\ [A-Z][a-z]+", $_POST{"Nom"}))
    print "Nom OK<BR \>\n";
  Equation => 1 pt
  if (ereg("[0-9]+\ ([*+-/] [0-9]+)+ = [0-9]+", $_POST{"Equation"}))
    print "Equation OK<BR \>\n";
?>
```

Université d'Aix-Marseille I
23/06/2005

Outils de l'Internet
$2^{ème}$ session

Examen

Documents autorisés : Polycopié du cours, notes de cours
Durée : 3h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Ligue 1 de football (sur 6 pts)

Un journaliste de l'Equipe décide de faciliter ses analyses sur les résultats du championnat de football de ligue 1 en créant un script PHP. Les données qu'il souhaite utiliser sont représentées dans le journal sous la forme d'un tableau :

Equipe	**Pts**	**Journée**	**G**	**N**	**P**	**Bm**	**Be**	**Diff**
Marseille	70	24	23	1	0	61	10	51
Istres	65	24	21	2	1	45	25	20
Lens	45	23	12	9	2	23	14	9
Paris	3	23	0	3	20	1	52	-51
Monaco	52	24	15	7	2	37	36	1
Auxerre	22	24	7	1	16	28	32	-4
...	...	...	...	...	...	...	...	...

où G = nombre de matchs gagnés, N = nombre de matchs nuls, P = nombre de matchs perdus, Bm = buts marqués, Be = buts encaissés, et Diff = différence de buts.
Ces informations sont récupérées sur le site de l'AFP sous la forme d'un fichier texte où chaque champs est séparé par un caractère '|' (Marseille|70|24|...). C'est donc ce fichier texte qui devra être lu pour récupérer les informations.

1.1 Lire le fichier (sur 0.5 pts)

Ecrire une fonction *lecture* prenant comme paramètre le nom du fichier et qui renverra le tableau dans lequel on aura stocké les données (table de hachage de la forme : \$tab{Equipe}{Pts}, \$tab{Equipe}{Journée}, ...).

1.2 Classement général (sur 0.5 pts)

Ecrire une fonction *classement* prenant comme paramètre la table contenant les données (cf. 1.1) et qui affiche le nom des équipes par ordre de Pts décroissant.

1.3 Lanterne rouge (sur 0.5 pts)

Ecrire une fonction *lanterne_rouge* prenant comme paramètre la table contenant les données et qui renvoie le nom de la plus mauvaise équipe du championnat (celle qui à le moins de Pts).

1.4 Vérification (sur 1 pt)

Ecrire une fonction *verification* prenant comme paramètre la table contenant les données et qui va vérifier que les calculs sont corrects (que Diff = Bm - Be, que Pts = G*3 + N, et que Journée = G + N + P). Si les calculs sont faux, la table sera corrigée.

1.5 Recherche de lettres dans le nom des équipes (sur 1 pt)

Ecrire une fonction *enumeration* prenant comme paramètres la table contenant les données et un caractère. La fonction affichera la liste des équipes classées par ordre décroissant d'occurences du caractère dans le nom de chacune de ces équipes. Par exemple, si l'on recherche le caractère "e", on obtiendra :

```
Marseille -> 2
Auxerre -> 2
Istres -> 1
...
```

1.6 JPGraph ou FPDF (sur 1.5 pts)

Choisir et traiter une (et une seule des deux questions suivantes) :

- **JPGraph** : Ecrire un script qui lira un fichier grâce à la fonction *lecture* (cf. 1.1) et génèrera un diagramme camembert en 3D (jpgraph_pie3d) des buts marqués par chaque équipe.
- **FPDF** : Ecrire un script qui lira un fichier grâce à la fonction *lecture* et génèrera un document PDF contenant le tableau des résultats de ligue 1.

2. Le code inconnu (sur 6.5 pts)

Le code suivant (*"test.php"*) a été récupéré sur internet. Complétez-le, corrigez-le, et expliquez ce qu'il fait.

```
1  <!DTD HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            // Début du code PHP
9                function m($a, $b)
10               {
11                    $res=$a*$b;
12                    ..............
13               }
14               ..................
15               {
16                   print "<TABLE>\n";
17                   foreach ($T as .......)
18                       foreach ($tab2 as $cle2 => $val)
19                           print "<TR><TD>$cle</TD><TD>".m($cle2, $val)."</TD>\n";
20                   print "</TABLE>\n";
21               }
22               if (empty($_POST{"Passe"}))
23               {
24                   print "<FORM action="test.php" method="POST">\n";
25                   print "A1 = <INPUT type="text" name="A1" />\n";
26                   print "A2 = <INPUT type="text" name="A2" />\n";
27                   print "<INPUT type="submit" value="Ok" />\n";
28                   ....................................
29                   print "</FORM>\n";
30               }
31               else
32               {
33                   for ($i=0; $i<50; $i++)
34                       $table{$i}{$_POST{"A1"}}=m($_POST{"A2"}, $i);
35                   aff($table);
36               }
37           PHP?>
38      </BODY>
39 </HTML>
```

3. Un peu de XML (sur 3 pts)

Ecrire un script permettant de lire un fichier *"fic.txt"* ayant la structure suivante (la première ligne est donnée à titre indicatif, elle ne fait pas partie du fichier) :

```
Nom|Prénom|Adresse|Code Postal|Ville
Geller|Ross|39 rue Joliot Curie|13013|Marseille
Buffay|Phoebe|10 rue Dutoit|75005|Paris
Geller|Monica|5 bd Berlioz|13003|Marseille
Bing|Chandler|12 av du Temple|92005|Malakoff
```

Une fois le fichier lu, le stocker dans une structure appropriée, et l'afficher sous forme XML en l'ayant trié par ordre alphabétique sur les nom et prénom des personnages. Voici un exemple de sortie attendue :

```
<PERSONNAGE>
  <NOM>
    Geller
  </NOM>
  <PRENOM>
    Ross
  </PRENOM>
  <ADRESSE CP="13013">
    39 rue Joliot Curie
  </ADRESSE>
  <VILLE>
    Marseille
  </VILLE>
</PERSONNAGE>
```

Enfin, donner la DTD de ce fichier (l'élément racine pourra être *FRIENDS*).

4. Un compteur de visiteurs (sur 2.5 pts)

Ecrire un script permettant de compter le nombre de passages d'un visiteur sur une page. Attention, l'utilisateur ne sera compté qu'une fois par jour !
Vous êtes libre d'utiliser les structures que vous désirez, base de données, ... à condition de l'expliquer ...

5. Base de données "Notes d'examen" (sur 2 pts)

On dispose d'un compte sur un serveur MySQL (Nom : "LicPro", Mot de passe : "20àl'exam", Serveur : "CMI") et on travaille sur la base "Examen" qui contient une table "RESULTATS" contenant les champs : Nom, Prenom, Exam_PHP, Projet_PHP, Final_PHP (seul le champs Final_PHP est vide). Sachant que la note finale de l'examen sera calculée comme suit :
`Final_PHP=(3*Exam_PHP+Projet_PHP)/4`
Créez un script permettant de se connecter à la base et de remplir le champs Final_PHP.

Université d'Aix-Marseille I
23/06/2005

Outils de l'Internet
$2^{ème}$ session

Examen (corrigé)

Documents autorisés : Polycopié du cours, notes de cours
Durée : 3h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Ligue 1 de football (sur 6 pts)

1.1 Lire le fichier (sur 0.5 pts)

```
function lecture($fichier)
{
  $fic=fopen($fichier, "r");
  if (empty($fic))
  {
    print "Erreur lors de l'ouverture de $fichier<BR />\n";
    exit;
  }
  while (!feof($fic))
  {
    $ligne=fgets($fic, 1024);
    $tab_ligne=explode("|", $ligne);
    $tab{$tab_ligne[0]}{"Pts"}=$tab_ligne[1];
    $tab{$tab_ligne[0]}{"Journée"}=$tab_ligne[2];
    $tab{$tab_ligne[0]}{"G"}=$tab_ligne[3];
    $tab{$tab_ligne[0]}{"N"}=$tab_ligne[4];
    $tab{$tab_ligne[0]}{"P"}=$tab_ligne[5];
    $tab{$tab_ligne[0]}{"Bm"}=$tab_ligne[6];
    $tab{$tab_ligne[0]}{"Be"}=$tab_ligne[7];
    $tab{$tab_ligne[0]}{"Diff"}=$tab_ligne[8];
  }
  return($tab);
}
```

1.2 Classement général (sur 0.5 pts)

```
function classement($table)
{
  foreach ($table as $cle => $tab)
    $T{$cle}=$tab{"Pts"};
  arsort($T);
  foreach ($T as $cle => $val)
    print "$val => $cle<BR />\n";
}
```

– ou –

```
function classement($table)
{
  arsort($table);
  foreach ($table as $cle => $tab)
    print "$cle => $tab{"Pts"}<BR />\n";
}
```

1.3 Lanterne rouge (sur 0.5 pts)

```
function lanterne_rouge($table)
{
  $mini=-1;
  foreach ($table as $cle => $tab)
  {
    if ($tab{"Pts"} < $mini)
    {
      $mini=$tab{"Pts"};
      $lr=$cle;
    }
  }
  return($lr);
}
```

1.4 Vérification (sur 1 pt)

```
function verification($table)
{
  foreach ($table as $cle => $tab)
  {
    $diff=$tab{"Diff"};
    $Bm=$tab{"Bm"};
    $Be=$tab{"Be"};
```

```
    $Pts=$tab{"Pts"};
    $G=$tab{"G"};
    $P=$tab{"P"};
    $Jour=$tab{"Journée"};
    $table{$cle}{"Diff"}=$Bm-$Be;
    $table{$cle}{"Pts"}=3*$G+$N;
    $table{$cle}{"Journée"}=$G+$N+$P;
  }
  return($table);
}
```

1.5 Recherche de lettres dans le nom des équipes (sur 1 pt)

```
function enumeration($table, $car)
{
  foreach ($table as $nom => $tab)
  {
    ereg($car, $nom,  $TabOcc);
    $T{$nom}=count($TabOcc)-1;
  }
  arsort($T);
  foreach ($T as $cle => $val)
    print "$cle -> $val<BR />\n";
}
```

1.6 JPGraph ou FPDF (sur 1.5 pts)

Choisir et traiter une (et une seule des deux questions suivantes) :

- **JPGraph :**

```
<?PHP
  include("/usr/share/jpgraph/jpgraph.php");
  include("/usr/share/jpgraph/jpgraph_pie3d.php");

  $i=0;
  $table=lecture("fichier.txt");
  foreach ($table as $cle => $tab)
    $data[$i++]=$tab{"Bm"};
  $graph=new PieGraph(300, 200);
  $pie=new PiePlot($data);
  $pie->SetTheme("earth");
  $pie->Explode(array(0, 15, 15, 25, 15));
  $graph->Add($pie);
  $graph->Stroke();
?>
```

– **FPDF :**

```
<?PHP
  define(FPDF_FONTPATH, "/usr/share/fpdf/font/");
  require("/usr/share/fpdf/fpdf.php");

  $table=lecture("fichier.txt");
  $pdf=new FPDF("P", "mm", "A4");
  $pdf->AddPage();
  $pdf->SetFont("Arial", "B", "16");
  $pdf->Cell(50, 10, "Equipe", 1, 0, "C");
  $pdf->Cell(50, 10, "Pts", 1, 0, "C");
  $pdf->Cell(50, 10, "Journée", 1, 0, "C");
  $pdf->Cell(50, 10, "G", 1, 0, "C");
  $pdf->Cell(50, 10, "N", 1, 0, "C");
  $pdf->Cell(50, 10, "P", 1, 0, "C");
  $pdf->Cell(50, 10, "Bm", 1, 0, "C");
  $pdf->Cell(50, 10, "Be", 1, 0, "C");
  $pdf->Cell(50, 10, "Diff", 1, 0, "C");
  $pdf->Ln();
  $pdf->SetFont("Arial", "", "16");
  foreach ($table as $cle => $tab)
  {
    $pdf->Cell(50, 10, $cle, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"Pts"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"Journée"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"G"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"N"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"P"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"Bm"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"Be"}, 1, 0, "C");
    $pdf->Cell(50, 10, $tab{"Diff"}, 1, 0, "C");
    $pdf->Ln();
  }
  $pdf->OutPut();
?>
```

2. Le code inconnu (sur 6.5 pts)

Le code suivant (*"test.php"*) a été récupéré sur internet. Complétez-le, corrigez-le, et expliquez ce qu'il fait.

```
1  <!DTD HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
   <!-- Erreur <!DOCTYPE ... (0.25 pts) -->
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            // Début du code PHP
             <!-- Les remarques // sont des remarques PHP (0.25 pts) -->
             <!-- Manque la balise PHP ouvrante (0.25 pts) -->
9                function m($a, $b)
10               {
11                    $res=$a*$b;
12                    ............
                      // Manque return($res) (0.5 pts)
13               }
14               .................
                 // Manque la déclaration de la fonction
                 // function aff($T) (1 pt)
15               {
16                   print "<TABLE>\n";
17                   foreach ($T as .......)
                     // foreach ($T as $cle => $tab2) (0.5 pts)
18                       foreach ($tab2 as $cle2 => $val)
19                           print "<TR><TD>$cle</TD><TD>".m($cle2, $val)."</TD>\n";
                             // Manque la balise fermante </TR> (0.5 pts)
20                   print "</TABLE>\n";
21               }
22               if (empty($_POST{"Passe"}))
23               {
24                   print "<FORM action="test.php" method="POST">\n";
25                   print "A1 = <INPUT type="text" name="A1" />\n";
26                   print "A2 = <INPUT type="text" name="A2" />\n";
27                   print "<INPUT type="submit" value="Ok" />\n";
28                   ......................................
                     // Manque print "<INPUT type="hidden" name="Passe"
                     //               value="1" />\n"; (1.25 pt)
29                   print "</FORM>\n";
```

```
30              }
31              else
32              {
33                  for ($i=0; $i<50; $i++)
34                      $table{$i}{$_POST{"A1"}}=m($_POST{"A2"}, $i);
35                  aff($table);
36              }
37          PHP?>
            <!-- Mauvaise balise fermante ?> (0.25 pts) -->
38      </BODY>
39 </HTML>
```

Lors du premier passage, le script affiche un formulaire permettant de donner 2 nombres. Lors du second passage, le script crée une table de 50 valeurs (0.25 pts) contenant la table de multiplication 0xA2 à 49xA2 (0.5 pts) puis affiche sous forme de tableau HTML : 0 A1x0xA2 jusqu'à 49 A1x49xA2 (1 pt).

3. Un peu de XML (sur 3 pts)

```
<?PHP

  $fic=file("fic.txt");
  $t_fic=explode("\n", $fic);
  foreach ($t_fic as $ligne)
  {
    $tt_fic=explode("|", $ligne);
    $tab{$tt_fic[0]}{$tt_fic[1]}[0]=$tt_fic[2];
    $tab{$tt_fic[0]}{$tt_fic[1]}[0]=$tt_fic[3];
    $tab{$tt_fic[0]}{$tt_fic[1]}[0]=$tt_fic[4];
  }

  print "<HTML>\n";
  print "  <BODY>\n";
  ksort($tab);
  foreach ($tab as $nom => $tab2)
  {
    print "<PERSONNAGE><BR />\n";
    print "\t<NOM><BR />\n";
    print "\t\t$nom<BR />\n";
    print "\t</NOM><BR />\n";
    ksort($tab2);
    foreach ($tab2 as $prenom => $tab3)
    {
```

```
      print "\t<PRENOM><BR />\n";
      print "\t\t$prenom<BR />\n";
      print "\t</PRENOM><BR />\n";
      print "\t<ADRESSE CP=\"".$tab3[1]."\"><BR />\n";
      print "\t\t".$tab3[0]."\n";
      print "\t</ADRESSE>\n";
      print "\t<VILLE>\n"
      print "\t\t".$tab3[2]."\n";
      print "\t</VILLE>\n";
    }
    print "</PERSONNAGE><BR />\n";
  }
  print "  </BODY>\n";
  print "</HTML>\n";

?>
```

Et la DTD :

```
<!ELEMENT FRIENDS (PERSONNAGE+)>

<!ELEMENT PERSONNAGE (NOM, PRENOM, ADRESSE, VILLE)>
<!ELEMENT NOM (#PCDATA)>
<!ELEMENT PRENOM (#PCDATA)>
<!ELEMENT ADRESSE (#PCDATA)>
<!ATTLIST ADRESSE CP (#PCDATA)>
<!ELEMENT VILLE (#PCDATA)>
```

4. Un compteur de visiteurs (sur 2.5 pts)

```
<?PHP
  setcookie("Passe", "1", time()+24*3600);
  if (empty($_COOKIE{"Passe"}))
  {
    $connexion=mysql_pconnect("MYSQL_DREAMMACHINE", "CR_Grille", "CMI");
    if (empty($connexion))
    {
      print "Echec de la connexion ...<BR />\n";
      exit;
    }
    if (!mysql_select_db("MF", $connexion))
    {
      print "Accès à la base impossible ...<BR \>\n";
      exit;
```

```
    }
    $requete="SELECT Nbre_Visiteurs FROM MF.VISITEURS";
    $resultat=mysql_query($requete, $connexion);
    $elt=mysql_fetch_object($resultat);
    $N=$elt->Nbre_Visiteurs + 1;
    $requete="INSERT INTO MF.VISITEURS (Nbre_Visiteurs) VALUES ($N);";
    $resultat=mysql_query($requete, $connexion);
  }
  /**********************
       Suite du code
  **********************/
?>
```

5. Base de données "Notes d'examen" (sur 2 pts)

```
function Notes_Exam()
{
  $connexion=mysql_pconnect("CMI", "LicPro", "20àl'exam");
  if (empty($connexion))
  {
    print "Echec de la connexion ...<BR />\n";
    exit;
  }
  if (!mysql_select_db("Examen", $connexion))
  {
    print "Accès à la base impossible ...<BR \>\n";
    exit;
  }
  $resultat=mysql_query("SELECT Nom, Prenom, Exam_PHP, Projet_PHP FROM
                         EXAMEN.RESULTATS", $connexion);
  while ($elt=mysql_fetch_object($resultat))
  {
    $final=(3*$elt->Exam_PHP+Projet_PHP)/4;
    $requete="UPDATE EXAMEN.RESULTATS SET Final_PHP=\"".$final."\" WHERE
              Nom=\"".$elt->Nom."\" AND Prenom=\"".$elt->Prenom."\"";
    $res=mysql_query($requete, $connexion) or die(mysql_error());
  }
}
```

Université d'Aix-Marseille I — IP6 : BD Internet
26/09/2005 — $2^{ème}$ session

Examen

Documents autorisés : Polycopié du cours uniquement (pas de listing, etc.)
Durée : 2h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Conception de grilles de Mots-Fléchés (sur 10 pts)

Une société décide de produire en masse des grilles de mots-fléchés. Pour cela, elle va développer un logiciel en PHP permettant de générer des grilles à partir d'une base de données. Voici les informations dont vous disposez :

```
SERVEUR :          MYSQL_DREAMMACHINE
NOM DE LA BASE :   MF
LOGIN :            CR_Grille
PASSWORD :         CMI

NOM DE LA TABLE : MOTS_FLECHES
CHAMPS :
    - MOT (VARCHAR(30), PRIMARY KEY) : mot à découvrir
    - NBRE_LETTRES (INT)             : nombre de lettres du mot
    - DEF (VARCHAR(30))              : courte définition permettant de découvrir
                                       le mot
```

Par exemple,

MOT	NBRE_LETTRES	DEF
crues	5	grossières
ode	3	poésie lyrique
décapitées	10	étêtées
lie	3	fond de cuve
...	...	...

1.1. Fonction d'accès à la base (sur 0.25 pt)

Ecrire une fonction *connexion* permettant de se connecter à la base et renvoyant l'identifiant de connexion.

1.2. Importation d'un fichier texte (sur 0.75 pt)

Ecrire une fonction *import* prenant comme paramètre le nom d'un fichier texte (au format `MOT|NBRE_LETTRES|DEF` soit `crues|5|grossières` par exemple), qui insère ces données dans la base et les affiche à l'écran sous forme de tableau.

1.3. Tri des mots (sur 1 pt)

Ecrire une fonction *trier* qui affiche les mots de la base par ordre alphabétique croissant.

1.4. Recherche de lettres dans un mot (sur 3 pts)

Ecrire une fonction *enumeration* prenant comme paramètre une chaîne de caractères. La fonction affichera la liste des lettres (ainsi que leur nombre d'occurences) qui composent ce mot classées par ordre décroissant d'occurences. Par exemple, pour le mot *décapitées* :

```
é -> 2
a -> 1
c -> 1
d -> 1
e -> 1
i -> 1
p -> 1
s -> 1
t -> 1
```

1.5. Intersection entre mots (sur 3 pts)

Ecrire une fonction *intersection* prenant comme paramètres deux chaînes de caractères et qui renvoie dans une table les lettres communes aux deux chaînes (en commençant la recherche par les lettres de la première chaîne). Par exemple, avec *dé**c**apité**es*** et ***c**ru**es***, on obtiendrait le tableau :

```
$tableau[0]="c"
$tableau[1]="e"
$tableau[2]="s"
```

1.6. Création d'une grille en PDF (sur 2 pts)

Nous supposerons ici que nous disposons d'un tableau T contenant les mots de la grille et d'une table de hachage D contenant toutes les définitions concernant une grille de mots fléchés (de 20 x 30 lettres) :

```
$T[i][j] => lettre de la case (i,j)
si $T[i][j] est vide, c'est qu'il s'agit d'une définition se trouvant dans la
table $D{"i,j"}
```

A partir de ces informations, générer un document PDF contenant une grille vide (avec les définitions mais sans les mots) et une grille de correction (avec les définitions et les mots).

2. Base de données "Notes d'examen" (sur 3.5 pts)

On dispose d'un compte sur un serveur MySQL (Nom : "LicPro", Mot de passe : "20àl'exam", Serveur : "CMI") et on travaille sur la base "Examen" qui contient une table "RESULTATS" contenant les champs : Nom, Prenom, Exam_PHP, Projet_PHP, Final_PHP (seul le champs Final_PHP est vide). Sachant que la note finale de l'examen sera calculée comme suit :
`Final_PHP=(3*Exam_PHP+Projet_PHP)/4`
Créez un script permettant de se connecter à la base et de remplir le champs Final_PHP.

La suite au dos de la page ....

3. Le code inconnu (sur 6.5 pts)

Le code suivant (*"test.php"*) a été récupéré sur internet. Complétez-le, corrigez-le, et expliquez ce qu'il fait.

```
1  <!DTD HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            // Début du code PHP
9                function m($a, $b)
10               {
11                    $res=$a*$b;
12                    ..............
13               }
14               ..................
15               {
16                   print "<TABLE>\n";
17                   foreach ($T as .......)
18                       foreach ($tab2 as $cle2 => $val)
19                           print "<TR><TD>$cle</TD><TD>".m($cle2, $val)."</TD>\n";
20                   print "</TABLE>\n";
21               }
22               if (empty($_POST{"Passe"}))
23               {
24                   print "<FORM action="test.php" method="POST">\n";
25                   print "A1 = <INPUT type="text" name="A1" />\n";
26                   print "A2 = <INPUT type="text" name="A2" />\n";
27                   print "<INPUT type="submit" value="Ok" />\n";
28                   ....................................
29                   print "</FORM>\n";
30               }
31               else
32               {
33                   for ($i=0; $i<50; $i++)
34                       $table{$i}{$_POST{"A1"}}=m($_POST{"A2"}, $i);
35                   aff($table);
36               }
37           PHP?>
38      </BODY>
39 </HTML>
```

Université d'Aix-Marseille I
26/09/2005

IP6 : BD Internet
$2^{ème}$ session

Examen (corrigé)

Documents autorisés : Polycopié du cours uniquement (pas de listing, etc.)
Durée : 3h
Remarques :
- Les exercices sont indépendants.
- Lorsqu'une impression écran (*print()* ou *echo()*) vous est demandée, on attendra une sortie au format HTML.

1. Conception de grilles de Mots-Fléchés (sur 10 pts)

1.1. Fonction d'accès à la base (sur 0.25 pt)

```
function connexion()
{
  $connexion=mysql_pconnect("MYSQL_DREAMMACHINE", "CR_Grille", "CMI");
  if (empty($connexion))
  {
    print "Echec de la connexion ...<BR />\n";
    exit;
  }
  if (!mysql_select_db("MF", $connexion))
  {
    print "Accès à la base impossible ...<BR \>\n";
    exit;
  }
  return($connexion);
}
```

1.2. Importation d'un fichier texte (sur 0.75 pt)

```
function import($fichier, $connexion)
{
  $f=fopen($fichier, "r");
  if (empty($f))
  {
    print "Impossible d'ouvrir le fichier $f ...<BR />\n";
    exit;
  }
```

```
  print "<TABLE>\n";
  print "<TR><TH>MOT</TH><TH>NBRE_LETTRES</TH><TH>DEF</TH></TR>\n";
  while (!feof($f))
  {
    $ligne=fgets($f, 1024);
    chop($ligne);
    $tab=explode("|", $ligne);
    $requete="INSERT INTO MF.MOTS_FLECHES (MOT, NBRE_LETTRES, DEF) VALUES
              ('".$tab[0]."', '".$tab[1]."', '". $tab[2]."');";
    $resultat=mysql_query($requete, $connexion);
    if (empty($resultat))
      die("Erreur d'exécution de la requête ...<BR />\n");
    print "<TR><TD>$tab[0]</TD><TD>$tab[1]</TD><TD>$tab[2]</TD></TR>\n";
  }
  print "</TABLE>\n";
}
```

1.3. Tri des mots (sur 1 pt)

```
function trier($connexion)
{
  $requete="SELECT MOT FROM MF.MOTS_FLECHES;";
  $resultat=mysql_query($requete, $connexion);
  while ($elt=mysql_fetch_object($resultat))
    $tab[$i++]=$elt->MOT;
  sort($tab);
  for ($j=0; $j<$i; $j++)
    print $tab[$j]."<BR />\n";
}
```

1.4. Recherche de lettres dans un mot (sur 3 pts)

```
function enumeration($chaine)
{
  $lng=count($chaine);
  for ($i=0; $i<$lng; $i++)
    $tab{$chaine[$i]}++;
  rsort($tab);
  foreach ($tab as $cle => $val)
    print "$val -> $cle<BR />\n";
}
```

1.5. Intersection entre mots (sur 3 pts)

```
function intersection($ch1, $ch2)
{
  $lg1=count($ch1);
  $lg2=count($ch2);
  for ($i=0; $i<$lg1; $i++)
  {
    for ($j=0; $j<$lg2; $j++)
      if ($ch1[$i] == $ch2[$j])
        $tab[$k++]=$ch1[$i];
  }
  return($tab);
}
```

1.6. Création d'une grille en PDF (sur 2 pts)

```
<?PHP
  define(FPDF_FONTPATH, "/usr/share/fpdf/font");
  require("/usr/share/fpdf/fpdf.php");
  $pdf=newFPDF("L", "mm", "A4");
  $pdf->AddPage();
  $pdf->SetFont("Arial", "B", "16");
  // Grille vide
  for ($i=0; $i<20; $i++)
  {
    for ($j=0; $j<30; $j++)
      if (empty($T[$i][$j]))
        $pdf->Cell(50, 10, $D{\"".$i.",".$j."\"}, 1, 1, "C");
      else
        $pdf->Cell(50, 10, "", 1, 1, "C");
    $pdf->Ln();
  }
  // Grille de correction
   for ($i=0; $i<20; $i++)
  {
    for ($j=0; $j<30; $j++)
      if (empty($T[$i][$j]))
        $pdf->Cell(50, 10, $D{\"".$i.",".$j."\"}, 1, 1, "C");
      else
        $pdf->Cell(50, 10, $T[$i][$j], 1, 1, "C");
    $pdf->Ln();
  }
  $pdf->OutPut();
```

```
?>
```

2. Base de données "Notes d'examen" (sur 3.5 pts)

```
function Notes_Exam()
{
  $connexion=mysql_pconnect("CMI", "LicPro", "20àl'exam");
  if (empty($connexion))
  {
    print "Echec de la connexion ...<BR />\n";
    exit;
  }
  if (!mysql_select_db("Examen", $connexion))
  {
    print "Accès à la base impossible ...<BR \>\n";
    exit;
  }
  $resultat=mysql_query("SELECT Nom, Prenom, Exam_PHP, Projet_PHP FROM
                         EXAMEN.RESULTATS", $connexion);
  while ($elt=mysql_fetch_object($resultat))
  {
    $final=(3*$elt->Exam_PHP+Projet_PHP)/4;
    $requete="UPDATE EXAMEN.RESULTATS SET Final_PHP=\"".$final."\" WHERE
              Nom=\"".$elt->Nom."\" AND Prenom=\"".$elt->Prenom."\"";
    $res=mysql_query($requete, $connexion) or die(mysql_error());
  }
}
```

3. Le code inconnu (sur 6.5 pts)

Le code suivant (*"test.php"*) a été récupéré sur internet. Complétez-le, corrigez-le, et expliquez ce qu'il fait.

```
1  <!DTD HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
   <!-- Erreur <!DOCTYPE ... (0.25 pts) -->
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4       <HEAD>
5            <TITLE>Petit programme PHP</TITLE>
6       </HEAD>
7       <BODY>
8            // Début du code PHP
             <!-- Les remarques // sont des remarques PHP (0.25 pts) -->
             <!-- Manque la balise PHP ouvrante (0.25 pts) -->
9                function m($a, $b)
10               {
11                    $res=$a*$b;
12                    .............
                      // Manque return($res) (0.5 pts)
13               }
14               .................
                 // Manque la déclaration de la fonction
                 // function aff($T) (1 pt)
15               {
16                   print "<TABLE>\n";
17                   foreach ($T as .......)
                     // foreach ($T as $cle => $tab2) (0.5 pts)
18                       foreach ($tab2 as $cle2 => $val)
19                           print "<TR><TD>$cle</TD><TD>".m($cle2, $val)."</TD>\n";
                             // Manque la balise fermante </TR> (0.5 pts)
20                   print "</TABLE>\n";
21               }
22               if (empty($_POST{"Passe"}))
23               {
24                   print "<FORM action="test.php" method="POST">\n";
25                   print "A1 = <INPUT type="text" name="A1" />\n";
26                   print "A2 = <INPUT type="text" name="A2" />\n";
27                   print "<INPUT type="submit" value="Ok" />\n";
28                   ......................................
                     // Manque print "<INPUT type="hidden" name="Passe"
                     //               value="1" />\n"; (1.25 pt)
29                   print "</FORM>\n";
```

```
30              }
31              else
32              {
33                  for ($i=0; $i<50; $i++)
34                      $table{$i}{$_POST{"A1"}}=m($_POST{"A2"}, $i);
35                  aff($table);
36              }
37          PHP?>
            <!-- Mauvaise balise fermante ?> (0.25 pts) -->
38      </BODY>
39 </HTML>
```

Lors du premier passage, le script affiche un formulaire permettant de donner 2 nombres. Lors du second passage, le script crée une table de 50 valeurs (0.25 pts) contenant la table de multiplication 0xA2 à 49xA2 (0.5 pts) puis affiche sous forme de tableau HTML : 0 A1x0xA2 jusqu'à 49 A1x49xA2 (1 pt).

Annexe E
Configuration de VIm avec PHP

Le répertoire `.vim/ftplugin` permet le stockage de fichiers de macros qui ne seront exécutés qu'en fonction du type de fichier. Ainsi, pour créer un fichier d'abbréviations spécifiques au PHP, on créera un fichier `.vim/ftplugin/php/abbrev.vim`. Il faut toutefois noter que, dans le cas du PHP, le fichier `.vim/ftplugin/html/abbrev.vim` sera également exécuté (ce qui est logique puisque le code PHP contiendra du code HTML enfoui). Voici donc deux exemples très simples de fichiers d'abréviations pour coder en PHP sous VIm. A vous de rajouter les abbréviations ou macros que vous jugerez utiles ...

.vim/ftplugin/html/abbrev.vim :

```
" Remplacement des accents par les codes HTML
iab é &eacute

iab è &egrave

iab à &agrave

" A l'écriture d'un Tag, affichage du tag fermant et repositionnement
" du curseur entre les deux tags
iab <TD> <TD></TD><Esc>4<Left>i
iab <td> <TD></TD><Esc>4<Left>i

iab <TR> <TR></TR><Esc>4<Left>i
iab <tr> <TR></TR><Esc>4<Left>i

" A l'écriture du Tag <IMG>, ajout des attributs classiques de ce tag
" et repositionnement du curseur entre les guillemets de l'attribut
" src
iab <IMG> <IMG src=\"\" alt=\"\" /><Esc>14<Left>i
iab <img> <IMG src=\"\" alt=\"\" /><Esc>14<Left>i

" Affichage du Tag fermant de <TABLE> et repositionnement du
" curseur
iab <TABLE> <TABLE>^M^M</TABLE>^[OA<TAB>
```

```
" Détermine un paragraphe où le texte sera centré
iab <CENTRE> <DIV align=\"center\">^M^M</DIV>^[OA<TAB>

" Affichage de la DTD
iab <DTD> <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
            "http://www.w3.org/TR/html4/loose.dtd">
iab <dtd> <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0.1 Transitional//EN"
            "http://www.w3.org/TR/html4/loose.dtd">
```

Ainsi, il suffit de taper `<IMG>_` (le caractère _ indique le curseur) suivi d'un caractère espace ou entrée pour obtenir
`<IMG src=\"_\" alt=\"\" />`

.vim/ftplugin/php/abbrev.vim :

```
" Formulaire POST
iab formpost <FORM method=\"POST\" action=\"\"><Esc>2<Left>i

" Formulaire GET
iab formget <FORM method=\"GET\" action=\"\"><Esc>2<Left>i

" Champs de formulaire : texte
iab inputtext <INPUT type=\"text\" name=\"\" size=\"\" value=\"\" />

" Champs de formulaire : bouton radio
iab inputradio <INPUT type=\"radio\" name=\"\" value=\"\" />

" Champs de formulaire : case à cocher
iab inputcheck <INPUT type=\"checkbox\" name=\"\" size=\"\" value=\"\" />
```

Annexe F
GNU Free Documentation Licence

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom : to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation : a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals ; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties : any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts : Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version :

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another ; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History" ; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http ://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any

later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM : How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page :

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation ; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this :

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Annexe G
GNU General Public Licence

Version 2, June 1991

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs ; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps : (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect

making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public License

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law : that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License ; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty ; keep intact all the notices that refer to this License and to the absence of any warranty ; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions :

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception : if the Program itself is interactive

but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you ; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following :

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange ; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange ; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims ; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system ; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix : How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty ; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

> ¡one line to give the program's name and a brief idea of what it does.¿
> Copyright (C) ¡year¿ ¡name of author¿
>
> This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode :

> Gnomovision version 69, Copyright (C) ¡year¿ ¡name of author¿
> Gnomovision comes with ABSOLUTELY NO WARRANTY ; for details type ‘show w’.
> This is free software, and you are welcome to redistribute it under certain conditions ; type ‘show c’ for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c` ; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample ; alter the names :

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> ¡signature of Ty Coon¿, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index

J

K

M

N

O

X

Outils de l'Internet
(côté Serveur)

La création de sites Web dynamiques, utilisant éventuellement des informations issues d'une base de données, nécessite la maîtrise d'un certain nombre d'outils et de langages. Le plus important d'entre eux permet de présenter des informations sur une page Web : il s'agit du HTML, langage de marquage statique, auquel on peut adjoindre des feuilles de style CSS permettant de modifier l'affichage d'une même page de code.
En dehors de cet aspect purement statique, il existe également des possibilités de programmation dynamique côté client ou côté serveur. La programmation côté client s'effectue essentiellement en JavaScript. On peut par la suite effectuer une combinaison d'HTML, CSS, et JavaScript ; on parlera alors de DHTML.
La programmation côté serveur peut être effectuée à l'aide de CGI (Common Gateway Interface) qui génèrent du code HTML à partir d'un programme en Perl ou C. Il existe également les JSP (Java Server Page) qui fonctionnent sur le même principe. Enfin, le langage le plus utilisé reste le PHP que l'on peut connecter très facilement à une base de données de type MySQL.
Pour finir, le XML permet le développement de nouveaux langages de marquages. Il a ainsi permit de définir de manière plus formelle le HTML qui devient alors du XHTML où les feuilles de style deviendront des XSLT.

Les différentes notions abordées (côté Client) :
- HTML : HyperText Markup Language
- Les feuilles de style en cascade : Cascading Style Sheets – CSS1 et CSS2
- JavaScript
- DHTML

Les différentes notions abordées (côté Serveur) :
- PHP : Hypertext Preprocessor
- PHP-MySQL
- Librairies PHP : FPDF et JPGraph
- XML : EXtensive Markup Language
- XSLT
- XHTML : EXtensive HyperText Markup Language

Toutes les notions présentées sont indépendantes du système d'exploitation utilisé. Toutefois, les explications seront toujours fournies en se référant au système Debian Linux, le lecteur pouvant l'adapter à son système.

LIF – 39 rue Joliot-Curie – 13453 Marseille Cedex 13