RECUEIL D'EXEMPLES PHP

Serge Tahé, université d'Angers, septembre 2001
serge.tahe@istia.univ-angers.fr

Avertissement

Ce document est un recueil de programmes PHP dont le but est de fournir des exemples d'utilisation de PHP dans différents domaines. Bien que PHP soit surtout connu pour son utilisation dans des environnements Web, je l'ai délibérément présenté en-dehors de ce contexte, tout simplement pour montrer que PHP est un langage semblable aux autres langages de scripts, notamment Perl, et que le Web n'est que l'un de ses champs d'application.

Ce document n'est en aucun cas un cours. Il est destiné à des gens sachant programmer et qui cherchent des exemples pour démarrer rapidement en PHP. Il nécessite un travail de lecture actif. Il s'agit en effet de découvrir ce qu'un programme fait et comment il le fait. Les programmes ont été commentés afin de vous y aider.

Le contenu du document devrait progressivement être enrichi de commentaires le rendant plus apte à être un support de cours.

Serge Tahé, septembre 2001

1 Les bases

1.1 Un premier exemple

Ci-dessous, on trouvera un programme présentant les premières caractéristiques de PHP.

1.1.1 Le programme

```
<?
  // ceci est un commentaire

  // variable utilisée sans avoir été déclarée
  $nom="dupont";
  // un affichage écran
  print "nom=$nom\n";
  // un tableau avec des éléments de type différent
  $tableau=array("un","deux",3,4);
  // son nombre d'éléments
  $n=count($tableau);
  // une boucle
  for($i=0;$i<$n;$i++)
    print "tableau[$i]=$tableau[$i]\n";
  // initialisation de 2 variables avec le contenu d'un tableau
  list($chaine1,$chaine2)=array("chaine1","chaine2");
  // concaténation des 2 chaînes
  $chaine3=$chaine1.$chaine2;
  // affichage résultat
  print "[$chaine1,$chaine2,$chaine3]\n";
  // utilisation fonction
  affiche($chaine1);
  // le type d'une variable peut être connu
  afficheType($n);
  afficheType($chaine1);
  afficheType($tableau);
  // le type d'une variable peut changer en cours d'exécution
  $n="a changé";
  afficheType($n);
  // une fonction peut rendre un résultat
  $res1=f1(4);
  print "res1=$res1\n";
  // une fonction peut rendre un tableau de valeurs
  list($res1,$res2,$res3)=f2();
  print "(res1,res2,res3)=[$res1,$res2,$res3]\n";
  // on aurait pu récupérer ces valeurs dans un tableau
  $t=f2();
  for($i=0;$i<count($t);$i++)
    print "t[$i]=$t[$i]\n";
  // des tests
  for($i=0;$i<count($t);$i++)
    // n'affiche que les chaînes
    if (getType($t[$i])=="string")
      print "t[$i]=$t[$i]\n";
  // d'autres tests
  for($i=0;$i<count($t);$i++)
    // n'affiche que les entiers >10
    if (getType($t[$i])=="integer" and $t[$i]>10)
      print "t[$i]=$t[$i]\n";
  // une boucle while
  $t=array(8,5,0,-2,3,4);
  $i=0;
  $somme=0;
  while($i<count($t) and $t[$i]>0){
    print "t[$i]=$t[$i]\n";
    $somme+=$t[$i];    //$somme=$somme+$t[$i]
    $i++;              //$i=$i+1
  }//while
  print "somme=$somme\n";

  // fin programme
  exit;
```

```php
  //----------------------------------
  function affiche($chaine){
    // affiche $chaine
    print "chaine=$chaine\n";
  }//affiche

  //----------------------------------
  function afficheType($variable){
    // affiche le type de $variable
    print "type[$variable]=".getType($variable)."\n";
  }//afficheType

  //----------------------------------
  function f1($param){
    // ajoute 10 à $param
    return $param+10;
  }

  //----------------------------------
  function f2(){
    // rend 3 valeurs
    return array("un",0,100);
  }

?>
```

Les résultats écran sont les suivants :

```
nom=dupont
tableau[0]=un
tableau[1]=deux
tableau[2]=3
tableau[3]=4
[chaine1,chaine2,chaine1chaine2]
chaine=chaine1
type[4]=integer
type[chaine1]=string
type[Array]=array
type[a changé]=string
res1=14
(res1,res2,res3)=[un,0,100]
t[0]=un
t[1]=0
t[2]=100
t[0]=un
t[2]=100
t[0]=8
t[1]=5
somme=13
```

1.2 La portée des variables

1.2.1 Programme 1

```php
<?
  // portée des variables

  function f1(){
    // on utilise la variable globale $i
    global $i;
    $i++;
    $j=10;
    print "f1[i,j]=[$i,$j]\n";
  }//f1

  function f2(){
    // on utilise la variable globale $i
    global $i;
    $i++;
    $j=20;
    print "f2[i,j]=[$i,$j]\n";
  }//f1
```

```
    function f3(){
    // on utilise une variable locale $i
    $i++;
    $j=30;
    print "f3[i,j]=[$i,$j]\n";
  }//f1

  // tests
  $i=0;$j=0;  // ces deux variables ne sont connues d'une fonction f
              // que si celle-ci déclare explicitement par l'instruction global
              // qu'elle veut les utiliser
  f1();
  f2();
  f3();
  print "test[i,j]=[$i,$j]\n";
?>
```

Les résultats :

```
f1[i,j]=[1,10]
f2[i,j]=[2,20]
f3[i,j]=[1,30]
test[i,j]=[2,0]
```

1.2.2 Programme 2

```
<?
  // la portée d'une variable est globale aux blocs de code
  $i=0;
  {
    $i++;
  }
  print "i=$i\n";
?>
```

Les résultats :

```
i=1
```

1.3 Les tableaux

1.3.1 Tableaux classiques à une dimension

```
<?
  // tableaux classiques
  // initialisation
  $tab1=array(0,1,2,3,4,5);
  // parcours - 1
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";
  // parcours - 2
  print "tab1 a " . count($tab1) . " éléments\n";
  reset($tab1);
  while(list($clé,$valeur)=each($tab1))
    print "tab1[$clé]=$valeur\n";
  // ajout d'éléments
  $tab1[]=$i++;
  $tab1[]=$i++;
  // parcours - 3
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";
  // suppression dernier élément
  array_pop($tab1);
  // parcours - 4
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";
  // suppression premier élément
```

```php
  array_shift($tab1);
  // parcours - 5
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";
  // ajout en fin de tableau
  array_push($tab1,-2);
  // parcours - 6
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";
  // ajout en début de tableau
  array_unshift($tab1,-1);
  // parcours - 7
  print "tab1 a " . count($tab1) . " éléments\n";
  for($i=0;$i<count($tab1);$i++)
    print "tab1[$i]=$tab1[$i]\n";

?>
```

Les résultats :

```
tab1 a 6 éléments
tab1[0]=0
tab1[1]=1
tab1[2]=2
tab1[3]=3
tab1[4]=4
tab1[5]=5
tab1 a 6 éléments
tab1[0]=0
tab1[1]=1
tab1[2]=2
tab1[3]=3
tab1[4]=4
tab1[5]=5
tab1 a 8 éléments
tab1[0]=0
tab1[1]=1
tab1[2]=2
tab1[3]=3
tab1[4]=4
tab1[5]=5
tab1[6]=6
tab1[7]=7
tab1 a 7 éléments
tab1[0]=0
tab1[1]=1
tab1[2]=2
tab1[3]=3
tab1[4]=4
tab1[5]=5
tab1[6]=6
tab1 a 6 éléments
tab1[0]=1
tab1[1]=2
tab1[2]=3
tab1[3]=4
tab1[4]=5
tab1[5]=6
tab1 a 7 éléments
tab1[0]=1
tab1[1]=2
tab1[2]=3
tab1[3]=4
tab1[4]=5
tab1[5]=6
tab1[6]=-2
tab1 a 8 éléments
tab1[0]=-1
tab1[1]=1
tab1[2]=2
tab1[3]=3
tab1[4]=4
tab1[5]=5
tab1[6]=6
```

```
tab1[7]=-2
```

1.3.2 Le dictionnaire

```
<?
  // dictionnaires
  $conjoints=array("Pierre"=>"Gisèle", "Paul"=>"Virginie", "Jacques"=>"Lucette","Jean"=>"");
  // parcours - 1
  print "Nombre d'éléments du dictionnaire : " . count($conjoints). "\n";
  reset($conjoints);
  while(list($clé,$valeur)=each($conjoints))
    print "conjoints[$clé]=$valeur\n";
  // tri du dictionnaire sur la clé
  ksort($conjoints);
  // parcours - 2
  reset($conjoints);
  while(list($clé,$valeur)=each($conjoints))
    print "conjoints[$clé]=$valeur\n";
  // liste des clés du dictionaire
  $clés=array_keys($conjoints);
  for($i=0;$i<count($clés);$i++)
    print "clés[$i]=$clés[$i]\n";
  // liste des valeurs du dictionaire
  $valeurs=array_values($conjoints);
  for($i=0;$i<count($valeurs);$i++)
    print "valeurs[$i]=$valeurs[$i]\n";
  // recherche d'une clé
  existe($conjoints,"Jacques");
  existe($conjoints,"Lucette");
  existe($conjoints,"Jean");
  // suppression d'une clé-valeur
  unset ($conjoints["Jean"]);
  print "Nombre d'éléments du dictionnaire : " . count($conjoints). "\n";
  reset($conjoints);
  while(list($clé,$valeur)=each($conjoints))
    print "conjoints[$clé]=$valeur\n";
  // fin
  exit;

 function existe($conjoints,$mari){
  // vérifie si la clé $mari existe dans le dictionnaire $conjoints
    if(isset($conjoints[$mari]))
      print "La clé [$mari] existe associée à la valeur [$conjoints[$mari]]\n";
    else print "La clé [$mari] n'existe pas\n";
  }

?>
```

Les résultats :

```
Nombre d'éléments du dictionnaire : 4
conjoints[Pierre]=Gisèle
conjoints[Paul]=Virginie
conjoints[Jacques]=Lucette
conjoints[Jean]=
conjoints[Jacques]=Lucette
conjoints[Jean]=
conjoints[Paul]=Virginie
conjoints[Pierre]=Gisèle
clés[0]=Jacques
clés[1]=Jean
clés[2]=Paul
clés[3]=Pierre
valeurs[0]=Lucette
valeurs[1]=
valeurs[2]=Virginie
valeurs[3]=Gisèle
La clé [Jacques] existe associée à la valeur [Lucette]
La clé [Lucette] n'existe pas
La clé [Jean] existe associée à la valeur []
Nombre d'éléments du dictionnaire : 3
conjoints[Jacques]=Lucette
conjoints[Paul]=Virginie
```

```
conjoints[Pierre]=Gisèle
```

1.3.3 Les tableaux à plusieurs dimensions

```
<?
  // tableaux classiques multidimensionnels
  // initialisation
  $multi=array(array(0,1,2), array(10,11,12,13), array(20,21,22,23,24));
  // parcours
  for ($i1=0;$i1<count($multi);$i1++)
    for ($i2=0;$i2<count($multi[$i1]);$i2++)
      print "multi[$i1][$i2]=".$multi[$i1][$i2]."\n";
  // dictionnaires multidimensionnels
  // initialisation
  $multi=array("zéro"=>array(0,1,2), "un"=>array(10,11,12,13), "deux"=>array(20,21,22,23,24));
  // parcours
  reset($multi);
  while(list($clé,$valeur)=each($multi))
    for ($i2=0;$i2<count($valeur);$i2++)
      print "multi[$clé][$i2]=".$multi[$clé][$i2]."\n";

?>
```

Résultats :

```
multi[0][0]=0
multi[0][1]=1
multi[0][2]=2
multi[1][0]=10
multi[1][1]=11
multi[1][2]=12
multi[1][3]=13
multi[2][0]=20
multi[2][1]=21
multi[2][2]=22
multi[2][3]=23
multi[2][4]=24
multi[zéro][0]=0
multi[zéro][1]=1
multi[zéro][2]=2
multi[un][0]=10
multi[un][1]=11
multi[un][2]=12
multi[un][3]=13
multi[deux][0]=20
multi[deux][1]=21
multi[deux][2]=22
multi[deux][3]=23
multi[deux][4]=24
```

1.3.4 Liens entre chaînes et tableaux

```
<?
  // chaîne vers tableau
  $chaine="1:2:3:4";
  $tab=explode(":",$chaine);
  // parcours tableau
  print "tab a " . count($tab) . " éléments\n";
  for($i=0;$i<count($tab);$i++)
    print "tab[$i]=$tab[$i]\n";
  // tableau vers chaîne
  $chaine2=implode(":",$tab);
  print "chaine2=$chaine2\n";
  // ajoutons un champ vide
  $chaine.=":";
  print "chaîne=$chaine\n";
  $tab=explode(":",$chaine);
  // parcours tableau
  print "tab a " . count($tab) . " éléments\n";
  for($i=0;$i<count($tab);$i++)
    print "tab[$i]=$tab[$i]\n"; // on a maintenant 5 éléments, le dernier étant vide
  // ajoutons de nouveau un champ vide
```

```
  $chaine.=":";
  print "chaîne=$chaine\n";
  $tab=explode(":",$chaine);
  // parcours tableau
  print "tab a " . count($tab) . " éléments\n";
  for($i=0;$i<count($tab);$i++)
    print "tab[$i]=$tab[$i]\n"; // on a maintenant 6 éléments, les deux derniers étant vides
?>
```

Résultats :

```
tab a 4 éléments
tab[0]=1
tab[1]=2
tab[2]=3
tab[3]=4
chaine2=1:2:3:4
chaîne=1:2:3:4:
tab a 5 éléments
tab[0]=1
tab[1]=2
tab[2]=3
tab[3]=4
tab[4]=
chaîne=1:2:3:4::
tab a 6 éléments
tab[0]=1
tab[1]=2
tab[2]=3
tab[3]=4
tab[4]=
tab[5]=
```

1.4 Les nombres réels

Php4/win9x présente un bogue de gestion des nombres réels.

```
<?
  // réel1 sera incorrect, les décimales étant ignorées
  $réel1=3.4+1.1;
  print "réel1=$réel1\n"; //affiche 4
  // réel2=3,4+1,1 n'est pas accepté
  // réel2="3,4"+"1,1" est accepté et mémorise la bonne valeur
  $réel2="3,4"+"1,1";
  print "réel2=$réel2\n"; //affiche 4,5
  // réel3 et réel4 sont corrects
  $réel3="2,2";
  $réel4=$réel2*$réel3;
  print "réel4=$réel4\n"; // correct : 9,9
?>
```

Résultats :

```
réel1=4
réel2=4,5
réel4=9,9
```

1.5 Les chaînes de caractères

1.5.1 Notation

```
<?
  // notation des chaînes
  $chaine1="un";
  $chaine2='un';
  $chaine3=un;
  print "[$chaine1,$chaine2,$chaine3]\n";
```

```
?>
```

Résultats :

```
[un,un,un]
```

1.5.2 Comparaison

```
<?
  // tests de comparaisons de chaînes
  compare("abcd","abcd");
  compare("","");
  compare("1","");
  compare("un",un);
  exit;

  function compare($chaine1,$chaine2){
    // compare chaine1 et chaine2
    if ($chaine1 == $chaine2)
      print "[$chaine1] est égal à [$chaine2]\n";
    else
      print "[$chaine1] est différent de [$chaine2]\n";
  }//compare
?>
```

Résultats :

```
[abcd] est égal à [abcd]
[] est égal à []
[1] est différent de []
[un] est égal à [un]
```

1.6 Les expressions régulières

```
<?
  // expression régulières en php
  // récupérer les différents champs d'une chaîne
    // le modèle : une suite de chiffres entourée de caractères quelconques
    // on ne veut récupérer que la suite de chiffres
    $modèle="/(\d+)/";
    // on confronte la chaîne au modèle
    @compare($modèle,"xyz1234abcd");
    @compare($modèle,"12 34");
    @compare($modèle,"abcd");

    // le modèle : une suite de chiffres entourée de caractères quelconques
    // on veut la suite de chiffres ainsi que les champs qui suivent et précèdent
    $modèle="/^(.*?)(\d+)(.*?)$/";
    // on confronte la chaîne au modèle
    @compare($modèle,"xyz1234abcd");
    @compare($modèle,"12 34");
    @compare($modèle,"abcd");

    // le modèle - une date au format jj/mm/aa
    $modèle="/^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$/";
    @compare($modèle,"10/05/97");
    @compare($modèle,"  04/04/01  ");
    @compare($modèle,"5/1/01");

    // le modèle - un nombre décimal
    $modèle="/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/";
    compare($modèle,"187.8");
    compare($modèle,"-0.6");
    compare($modèle,"4");
    compare($modèle,".6");
    compare($modèle,"4.");
    compare($modèle," + 4");

    // fin
```

```
    exit;

  // ------------------------------------------------------------------------
    function compare($modèle,$chaîne){
      // compare la chaîne $chaîne au modèle $modèle
    // on confronte la chaîne au modèle
    $correspond=preg_match($modèle,$chaîne,$champs);
    // affichage résultats
    print "\nRésultats($modèle,$chaîne)\n";
    if($correspond){
      for($i=0;$i<count($champs);$i++)
        print "champs[$i]=$champs[$i]\n";
    } else print "La chaîne [$chaîne] ne correspond pas au modèle [$modèle]\n";
  }//compare

?>
```

Résultats :

```
Résultats(/(\d+)/,xyz1234abcd)
champs[0]=1234
champs[1]=1234

Résultats(/(\d+)/,12 34)
champs[0]=12
champs[1]=12

Résultats(/(\d+)/,abcd)
La chaîne [abcd] ne correspond pas au modèle [/(\d+)/]

Résultats(/^(.*?)(\d+)(.*?)$/,xyz1234abcd)
champs[0]=xyz1234abcd
champs[1]=xyz
champs[2]=1234
champs[3]=abcd

Résultats(/^(.*?)(\d+)(.*?)$/,12 34)
champs[0]=12 34
champs[1]=
champs[2]=12
champs[3]= 34

Résultats(/^(.*?)(\d+)(.*?)$/,abcd)
La chaîne [abcd] ne correspond pas au modèle [/^(.*?)(\d+)(.*?)$/]

Résultats(/^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$/,10/05/97)
champs[0]=10/05/97
champs[1]=10
champs[2]=05
champs[3]=97

Résultats(/^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$/,  04/04/01  )
champs[0]=  04/04/01
champs[1]=04
champs[2]=04
champs[3]=01

Résultats(/^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$/,5/1/01)
La chaîne [5/1/01] ne correspond pas au modèle [/^\s*(\d\d)\/(\d\d)\/(\d\d)\s*$/]

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/,187.8)
champs[0]=187.8
champs[1]=
champs[2]=187.8

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/,-0.6)
champs[0]=-0.6
champs[1]=-
champs[2]=0.6

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/,4)
champs[0]=4
champs[1]=
champs[2]=4

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/,.6)
```

```
champs[0]=.6
champs[1]=
champs[2]=.6

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/,4.)
champs[0]=4.
champs[1]=
champs[2]=4.

Résultats(/^\s*([+|-]?)\s*(\d+\.\d*|\.\d+|\d+)\s*/, + 4)
champs[0]= + 4
champs[1]=+
champs[2]=4
```

1.7 Mode de passage des paramètres des fonctions

```
<?
  // mode de pasage des paramètres

  function f(&$i,$j){
    // $i sera toujours obenu par référence quelque soit le mode passage du paramètre effectif
    // $j sera obtenu par référence seulement si le paramètre effectif est passé par référence
    // sinon $j sera obtenu par valeur
    $i++;
    $j++;
    print "f[i,j]=[$i,$j]\n";
  }//f

  // tests
  $i=0;$j=0;
  // $i et $j passés par valeurs
  f($i,$j);
  print "test[i,j]=[$i,$j]\n";
  // $i passé par référence, $j par valeur
  f(&$i,$j);
  print "test[i,j]=[$i,$j]\n";
  // $i passé par valeur, $j par référence
  f($i,&$j);
  print "test[i,j]=[$i,$j]\n";
  // $i passé par référence, $j par référence
  f(&$i,&$j);
  print "test[i,j]=[$i,$j]\n";
?>
```

Résultats :

```
f[i,j]=[1,1]
test[i,j]=[1,0]
f[i,j]=[2,1]
test[i,j]=[2,0]
f[i,j]=[3,1]
test[i,j]=[3,1]
f[i,j]=[4,2]
test[i,j]=[4,2]
```

1.8 Résultats rendus par une fonction

```
<?
  // résultats rendus par une fonction

  // une fonction peut rendre plusieurs valeurs dans un tableau
  list($res1,$res2,$res3)=f1(10);
  print "[$res1,$res2,$res3]\n";
  $res=f1(10);
  for($i=0;$i<count($res);$i++)
    print "res[$i]=$res[$i]\n";

  // une fonction peut rendre un objet
  $res=f2(10);
```

```php
  print "[$res->res1,$res->res2,$res->res3]\n";

  // fin
  exit;

  // fonction f1
  function f1($valeur){
    // rend un tableau ($valeur+1,$aleur+2,$baleur+3)
    return array($valeur+1,$valeur+2,$valeur+3);
  }//f1

  // fonction f2
  function f2($valeur){
    // rend un objet ($valeur+1,$aleur+2,$baleur+3)
    $res->res1=$valeur+1;;
    $res->res2=$valeur+2;
    $res->res3=$valeur+3;
    // rend l'objet
    return $res;
  }//f2
?>
```

Résultats

```
[11,12,13]
res[0]=11
res[1]=12
res[2]=13
[11,12,13]
```

Commentaires

- le programme précédent qu'une fonction php peut rendre un ensemble de résultats et non un seul, sous la forme d'un tableau ou d'un objet
- la notion d'objet est explicitée un peu plus loin

1.9 Les fichiers texte

```php
<?
  // exploitation séquentielle d'un fichier texte
  // celui-ci est un ensemble de lignes de la forme login:pwd:uid:gid:infos:dir:shell
  // chaque ligne est mis dans un dictionnaire sous la forme login => uid:gid:infos:dir:shell
  // on fixe le nom du fichier
  $INFOS="infos.txt";
  // on l'ouvre en création
  if (! $fic=@fopen($INFOS,"w")){
    print "Erreur d'ouverture du fichier $INFOS en écriture\n";
    exit;
  }
  // on génère un contenu arbitraire
  for($i=0;$i<100;$i++)
    fputs($fic,"login$i:pwd$i:uid$i:gid$i:infos$i:dir$i:shell$i\n");
  // on ferme le fichier
  fclose($fic);

  // on l'exploite - fgets garde la marque de fin de ligne
  // cela permet de ne pas récupérer une chaîne vide lors de la lecture d'une ligne blanche
  // on l'ouvre en création
  if (! $fic=@fopen($INFOS,"r")){
    print "Erreur d'ouverture du fichier $INFOS en lecture\n";
    exit;
  }
  while($ligne=fgets($fic,1000)){
    // on supprime la marque de fin de ligne si elle existe
    $ligne=cutNewLineChar($ligne);
    // on met la ligne dans un tableau
    $infos=explode(":",$ligne);
    // on récupère le login
    $login=array_shift($infos);
```

```
    // on néglige le pwd
    array_shift($infos);
    // on crée une entrée dans le dictionnaire
    $dico[$login]=$infos;
  }
  // on le ferme
  fclose($fic);

  // exploitation du dictionnaire
  afficheInfos($dico,"login10");
  afficheInfos($dico,"X");

  // fin
  exit;

  // ------------------------------------------------------------------------
  function afficheInfos($dico,$clé){
    // affiche la valeur associée à clé dans le dictionnaire $dico si elle existe
    $valeur=$dico[$clé];
    if (isset($valeur))
      // valeur existe - est-ce un tableau ?
      if (is_array($valeur))
        print "[$clé," . join(":",$valeur) ."]\n";
      // $valeur n'est pas un tableau
      else print "[$clé,erreur]\n";
    // $clé n'est pas une clé du dictionnaire $dico
    else print "la clé [$clé] n'existe pas\n";
  }//afficheInfos

  // ------------------------------------------------------------------------
  function cutNewLinechar($ligne){
  // on supprime la marque de fin de ligne de $ligne si elle existe
    $L=strlen($ligne);  // longueur ligne
    while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
      $ligne=substr($ligne,0,$L-1);
      $L--;
    }
    // fin
    return($ligne);
  }//cutNewLineChar

?>
```

Le fichier infos.txt :

```
login0:pwd0:uid0:gid0:infos0:dir0:shell0
login1:pwd1:uid1:gid1:infos1:dir1:shell1
login2:pwd2:uid2:gid2:infos2:dir2:shell2
...
login98:pwd98:uid98:gid98:infos98:dir98:shell98
login99:pwd99:uid99:gid99:infos99:dir99:shell99
```

Les résultats écran :

```
[login10,uid10:gid10:infos10:dir10:shell10]
la clé [X] n'existe pas
```

2 Exercice d'application - IMPOTS

2.1 Le problème

On se propose d'écrire un programme permettant de calculer l'impôt d'un contribuable. On se place dans le cas simplifié d'un contribuable n'ayant que son seul salaire à déclarer :

- on calcule le nombre de parts du salarié **nbParts=nbEnfants/2 +1** s'il n'est pas marié, **nbEnfants/2+2** s'il est marié, où *nbEnfants* est son nombre d'enfants.
- on calcule son revenu imposable **R=0.72*S** où S est son salaire annuel
- on calcule son coefficient familial **Q=R/N**
  on calcule son impôt I d'après les données suivantes

```
12620.0     0       0
13190       0.05    631
15640       0.1     1290.5
24740       0.15    2072.5
31810       0.2     3309.5
39970       0.25    4900
48360       0.3     6898.5
55790       0.35    9316.5
92970       0.4     12106
127860      0.45    16754.5
151250      0.50    23147.5
172040      0.55    30710
195000      0.60    39312
0           0.65    49062
```

Chaque ligne a 3 champs. Pour calculer l'impôt I, on recherche la première ligne où *QF<=champ1*. Par exemple, si QF=30000 on trouvera la ligne

24740 0.15 2072.5

L'impôt I est alors égal à **0.15*R - 2072.5*nbParts**. Si QF est tel que la relation *QF<=champ1* n'est jamais vérifiée, alors ce sont les coefficcients de la dernière ligne qui sont utilisés. Ici :

0 0.65 49062

ce qui donne l'impôt I=0.65*R - 49062*nbParts.

2.2 Version avec tableaux

```
<?
  // définition des constantes
  $DATA="data.txt";
  $RESULTATS="résultats.txt";
  $limites=array(12620,13190,15640,24740,31810,39970,48360,55790,92970,127860,151250,172040,19
5000);
  $coeffR=array("0","0,05","0,1","0,15","0,2","0,25","0,3","0,35","0,4","0,45","0,5","0,55","0
,6","0,65");
  $coeffN=array("0","631","1290,5","272,5","3309,5","4900","6898,5","9316,5","12106","16754,5"
,"23147,5","30710","39312","49062");

  // lecture des données
  $data=@fopen($DATA,"r");
  if(! $data){
    print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
    exit;
  }

  // ouverture fichier des résultats
  $résultats=@fopen($RESULTATS,"w");
  if(! $résultats){
```

```
    print "Impossible de créer le fichier des résultats [$RESULTATS]\n";
    exit;
  }

  // on exploite la ligne courante du fichier des données
  while($ligne=fgets($data,100)){
    // on enlève l'éventuelle marque de fin de ligne
    $ligne=cutNewLineChar($ligne);
    // on récupère les 3 champs marié:enfants:salaire qui forment $ligne
    list($marié,$enfants,$salaire)=explode(",",$ligne);
    // on calcule l'impôt
    $impôt=calculImpots($marié,$enfants,$salaire,$limites,$coeffR,$coeffN);
    // on inscrit le résultat
    fputs($résultats,"$marié:$enfants:$salaire:$impôt\n");
    // donnée suivante
  }// while

  // on ferme les fichiers
  fclose($data);
  fclose($résultats);

  // fin
  exit;

  // ----------------------------------------------------------------------
  function cutNewLinechar($ligne){
  // on supprime la marque de fin de ligne de $ligne si elle existe
    $L=strlen($ligne);  // longueur ligne
    while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
      $ligne=substr($ligne,0,$L-1);
      $L--;
    }
    // fin
    return($ligne);
  }//cutNewLineChar

  // ----------------------------------------------------------------------
  function calculImpots($marié,$enfants,$salaire,$limites,$coeffR,$coeffN){
    // $marié : oui, non
    // $enfants : nombre d'enfants
    // $salaire : salaire annuel
    // $limites, $coeffR, $coeffN : les tableaux des données permettant le calcul de l'impôt

    // nombre de parts
    $marié=strToLower($marié);
    if($marié=="oui") $nbParts=$enfants/2+2;
      else $nbParts=$enfants/2+1;
    // une 1/2 part de plus si au moins 3 enfants
    if($enfants>=3) $nbParts+="0,5";
    // revenu imposable
    $revenuImposable="0,72"*$salaire;
    // quotient familial
    $quotient=$revenuImposable/$nbParts;
    // est mis à la fin du tableau limites pour arrêter la boucle qui suit
    array_push($limites,$quotient);
    print "[$marié,$enfants,$salaire,$nbParts,$revenuImposable,$quotient]\n";
    // calcul de l'impôt
    $i=0;
    while($quotient>$limites[$i]) $i++;
    print "[$i,$coeffR[$i],$coeffN[$i]]\n";
    // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
    // ne peut déborder du tableau $limites
    // maintenant on peut calculer l'impôt
    return floor($revenuImposable*$coeffR[$i]-$nbParts*$coeffN[$i]);
  }//calculImpots
?>
```

Le fichier des données *data.txt* :

```
oui,2,200000
non,2,200000
oui,3,200000
non,3,200000
oui,5,50000
non,0,3000000
```

Les fichier *résultats.txt* des résultats obtenus :

```
oui:2:200000:22504
non:2:200000:33388
oui:3:200000:16400
non:3:200000:22504
oui:5:50000:0
non:0:3000000:1354938
```

2.3 Version avec fichiers texte

```
<?
  // définition des constantes
  $DATA="data.txt";
  $RESULTATS="résultats.txt";
  $IMPOTS="impots.txt";

  // les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
  // à raison d'une ligne par tableau sous la forme
  // val1:val2:val3,...
  list($erreur,$limites,$coeffR,$coeffN)=getTables($IMPOTS);

  // y-a-t-il eu une erreur ?
  if($erreur){
    print "$erreur\n";
    exit;
  }//if

  // lecture des données utilisateur
  $data=@fopen($DATA,"r");
  if(! $data){
    print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
    exit;
  }

  // ouverture fichier des résultats
  $résultats=@fopen($RESULTATS,"w");
  if(! $résultats){
    print "Impossible de créer le fichier des résultats [$RESULTATS]\n";
    exit;
  }

  // on exploite la ligne courante du fichier des données
  while($ligne=fgets($data,100)){
    // on enlève l'éventuelle marque de fin de ligne
    $ligne=cutNewLineChar($ligne);
    // on récupère les 3 champs marié:enfants:salaire qui forment $ligne
    list($marié,$enfants,$salaire)=explode(",",$ligne);
    // on calcule l'impôt
    $impôt=calculImpots($marié,$enfants,$salaire,$limites,$coeffR,$coeffN);
    // on inscrit le résultat
    fputs($résultats,"$marié:$enfants:$salaire:$impôt\n");
    // donnée suivante
  }// while

  // on ferme les fichiers
  fclose($data);
  fclose($résultats);

  // fin
  exit;

  // ------------------------------------------------------------------------
  function cutNewLinechar($ligne){
  // on supprime la marque de fin de ligne de $ligne si elle existe
    $L=strlen($ligne);  // longueur ligne
    while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
      $ligne=substr($ligne,0,$L-1);
      $L--;
    }
```

```
    // fin
    return($ligne);
  }//cutNewLineChar

  // ---------------------------------------------------------------------------
  function calculImpots($marié,$enfants,$salaire,$limites,$coeffR,$coeffN){
    // $marié : oui, non
    // $enfants : nombre d'enfants
    // $salaire : salaire annuel
    // $limites, $coeffR, $coeffN : les tableaux des données permettant le calcul de l'impôt

    // nombre de parts
    $marié=strToLower($marié);
    if($marié=="oui") $nbParts=$enfants/2+2;
      else $nbParts=$enfants/2+1;
    // une 1/2 part de plus si au moins 3 enfants
    if($enfants>=3) $nbParts+="0,5";
    // revenu imposable
    $revenuImposable="0,72"*$salaire;
    // quotient familial
    $quotient=$revenuImposable/$nbParts;
    // est mis à la fin du tableau limites pour arrêter la boucle qui suit
    array_push($limites,$quotient);
    // calcul de l'impôt
    $i=0;
    while($quotient>$limites[$i]) $i++;
    // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
    // ne peut déborder du tableau $limites
    // maintenant on peut calculer l'impôt
    return floor($revenuImposable*$coeffR[$i]-$nbParts*$coeffN[$i]);
  }//calculImpots

  // ---------------------------------------------------------------------------
  function getTables($IMPOTS){
    // $IMPOTS : le nom du fichier contenant les données des tables $limites, $coeffR, $coeffN
    // le fichier $IMPOTS existe-t-il ?
    if (! file_exists($IMPOTS)) return array("Le fichier $IMPOTS n'existe pas");
    // lecture en bloc du fichier
    $tables=file($IMPOTS);
    if (! $tables) return array("Erreur lors de l'exploitation du fichier $IMPOTS");
    // création des 3 tables - on suppose que les lignes sont syntaxiquement correctes
    $limites=explode(":",cutNewLineChar($tables[0]));
    $coeffR=explode(":",cutNewLineChar($tables[1]));
    $coeffN=explode(":",cutNewLineChar($tables[2]));
    // fin
    return array("",$limites,$coeffR,$coeffN);
  }//getTables
?>
```

Les résultats obtenus ici sont les mêmes qu'obtenus précédemment.

3 Les objets

3.1 Toute variable peut devenir un objet doté d'attributs

```
<?
  // toute variable peut avoir des attributs par construction
    $obj1->attr1="un";
    $obj1->attr2=100;
    // affiche l'objet
    print "objet1=[$obj1->attr1,$obj1->attr2]\n";
    // modifie l'objet
    $obj1->attr2+=100;
    // affiche l'objet
    print "objet1=[$obj1->attr1,$obj1->attr2]\n";
    // affecte la aleur de objet1 à objet2
    $obj2=$obj1;
    // modifie obj2
    $obj2->attr2=0;
    // affiche les deux objets
    print "objet1=[$obj1->attr1,$obj1->attr2]\n";
    print "objet2=[$obj2->attr1,$obj2->attr2]\n";
    // affecte la référence de objet1 à objet3
    $obj3=&$obj1;
    // modifie obj3
    $obj3->attr2=0;
    // affiche les deux objets
    print "objet1=[$obj1->attr1,$obj1->attr2]\n";
    print "objet3=[$obj3->attr1,$obj3->attr2]\n";
    // un objet est-il un tableau ?
    for($i=0;$i<count($obj1);$i++)
      print "obj1[$i]=$obj1[$i]\n";
    // un objet est-il un dictionnaire ?
    while(list($attribut,$valeur)=each($obj1))
      print "obj1[$attribut]=$valeur\n";
    // ajout d'une clé à l'objet obj1
    $obj1[attr1]="deux";
    print "$obj1->attr1\n";
    // inversement un dictionnaire peut-il être vu comme un objet ?
    $dico=array(attr1=>"un",attr2=>2);
    print "dico=[$dico->attr1,$dico->attr2]\n";
    print "dico=[$dico[attr1],$dico[attr2]]\n";
    // fin
    exit;
?>
```

Résultats :

```
objet1=[un,100]
objet1=[un,200]
objet1=[un,200]
objet2=[un,0]
objet1=[un,0]
objet3=[un,0]
obj1[0]=
obj1[1]=
obj1[attr1]=un
obj1[attr2]=0
<br>
<b>Warning</b>:  Cannot use a scalar value as an array in
<b>C:\data\serge\php\objets\1\php1031.php3</b> on line <b>31</b><br>
un
dico=[,]
dico=[un,2]
```

Commentaires

- $obj1 est un objet avec deux attributs. L'affichage nous montre que count($obj1) vaut 2 alors que la fonction count s'applique normalement à des tableaux. On pouvait alors

penser que l'objet $obj1 pouvait être vu aussi comme un tableau et que l'attribut i de l'objet $obj1 pouvait être obtenu par $obj1[$i]. La séquence de code

```
// un objet est-il un tableau ?
for($i=0;$i<count($obj1);$i++)
  print "obj1[$i]=$obj1[$i]\n";
// un objet est-il un dictionnaire ?
while(list($attribut,$valeur)=each($obj1))
  print "obj1[$attribut]=$valeur\n";
// ajout d'une clé à l'objet obj1
$obj1[attr1]="deux";
print "$obj1->attr1\n";
```

et les résultats associés

```
obj1[0]=
obj1[1]=
obj1[attr1]=un
obj1[attr2]=0
<br>
<b>Warning</b>:  Cannot use a scalar value as an array in
<b>C:\data\serge\php\objets\1\php1031.php3</b> on line <b>31</b><br>
un
```

montrent en fait que $obj1 peut être vu ni comme un tableau classique mais ni comme un dictionnaire. Cependant les fonctions **count** et **each** ont un sens sur un objet : elles ramènent respectivement le nombre d'attributs et la liste des paires (attribut,valeur) de l'objet.

- les variables présentées ici sont plus des structure au sens du langage C que des objets : elles n'ont en effet pas de méthodes, ni de moule (class) permettant de créer des objets semblables.

3.2 Une classe personne sans attributs déclarés

```
<?
  class personne{
    // attributs de la classe
    // non déclarés - peuvent être créés dynamiquement

    // méthode
    function identité(){
      // à priori, utilise des attributs inexistants
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité

  }//personne

  // test

  // les attributs sont publics et peuvent être créés dynamiquement
  $p=new personne;
  $p->prénom="Paul";
  $p->nom="Langevin";
  $p->âge=48;
  // appel d'une méthode
  print "personne=".$p->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
personne=[Paul,Langevin,48]
```

3.3 La classe personne avec attributs déclarés

Programme

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;

    // méthode
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité

  }//personne

  // test

  // les attributs sont publics
  $p=new personne;
  $p->prénom="Paul";
  $p->nom="Langevin";
  $p->âge=48;
  // appel d'une méthode
  print "personne=".$p->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
personne=[Paul,Langevin,48]
```

3.4 La classe personne avec un constructeur

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;

    // constructeur
    function personne($prénom,$nom,$âge){
      $this->prénom=$prénom;
      $this->nom=$nom;
      $this->âge=$âge;
    }//personne

    // méthode
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité

  }//personne

  // test

  // création d'un objet personne
  $p=new personne("Paul","Langevin",48);
  // identité de cette personne
  print "personne=".$p->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
personne=[Paul,Langevin,48]
```

3.5 La classe personne avec contrôle de validité dans le constructeur

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;
    var $erreur;

    // constructeur
    function personne($prénom,$nom,$âge){
      // le prénom doit être non vide
      if(! preg_match("/^\s*(.+?)\s*$/",$prénom,$champs)){
        $this->erreur=1;
        return;
      }else $this->prénom=$champs[1];
      // le nom doit être non vide
      if(! preg_match("/^\s*(.+?)\s*$/",$nom,$champs)){
        $this->erreur=2;
        return;
      }else $this->nom=$champs[1];
      // l'âge doit être valide
      if(! preg_match("/^\s*(\d+)\s*$/",$âge,$champs)){
        $this->erreur=3;
        return;
      }else $this->âge=$champs[1];
      // pas d'erreur
      $this->erreur=0;
    }//personne

    // méthode
    function identité(){
      switch ($this->erreur){
        case 0:
          return "[$this->prénom,$this->nom,$this->âge]";
          break;
        case 1:
          return "[erreur,$this->nom,$this->âge]";
          break;
        case 2:
          return "[$this->prénom,erreur,$this->âge]";
          break;
        case 3:
          return "[$this->prénom,$this->nom,erreur]";
          break;
      }//switch
    }//identité

  }//personne

  // test

  // création d'un objet personne
  $p=new personne("Paul","Langevin",48);
  // identité de cette personne
  print "personne=".$p->identité()."\n";
  // création d'un objet personne erroné
  $p=new personne("xx","yy","zz");
  print "personne=".$p->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
personne=[Paul,Langevin,48]
personne=[xx,yy,erreur]
```

3.6 Ajout d'une méthode faisant office de second constructeur

Programme

```php
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;
    var $erreur;

    // constructeur
    function personne($prénom,$nom,$âge){
      // au départ pas d'erreur
      $erreur=0;
      // le prénom doit être non vide
      if(! preg_match("/^\s*(.+?)\s*$/",$prénom,$champs)){
        $erreur+=1;
        $this->prénom="erreur";
      }else $this->prénom=$champs[1];
      // le nom doit être non vide
      if(! preg_match("/^\s*(.+?)\s*$/",$nom,$champs)){
        $erreur+=2;
        $this->nom="erreur";
      }else $this->nom=$champs[1];
      // l'âge doit être valide
      if(! preg_match("/^\s*(\d+)\s*$/",$âge,$champs)){
        $erreur+=4;
        $this->âge="erreur";
      }else $this->âge=$champs[1];
      // on note l'erreur
      $this->erreur=$erreur;
    }//personne

    // on ne peut mettre de second constructeur
    // function personne($p){
    // }

    function initWithPersonne($p){
      // initialise l'objet courant avec une personne $p
      $this->personne($p->prénom,$p->nom,$p->âge);
    }//initWithPersonne

    // méthode
    function identité(){
          return "[$this->prénom,$this->nom,$this->âge]";
    }//identité
  }//personne

  // test

  // création d'un objet personne
  $p=new personne("Paul","Langevin",48);
  // identité de cette personne
  print "personne=".$p->identité()."\n";
  // création d'un objet personne erroné
  $p=new personne("xx","yy","zz");
  print "personne=".$p->identité()."\n";
  // création d'une seconde personne
  $p2=new personne("Laure","Adeline",67);
  // initialisation de la première avec les valeurs de la seconde
  $p->initWithPersonne($p2);
  // vérification
  print "personne=".$p->identité()."\n";
  // provoquons une erreur
  $p->initWithPersonne("n'importe quoi");
  // vérification
  print "personne=".$p->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
personne=[Paul,Langevin,48]
personne=[xx,yy,erreur]
personne=[Laure,Adeline,67]
```

```
personne=[erreur,erreur,erreur]
```

3.7 Un tableau d'objets personne

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;

    // constructeur
    function personne($prénom,$nom,$âge){
      $this->prénom=$prénom;
      $this->nom=$nom;
      $this->âge=$âge;
    }//personne

    // méthode
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité

  }//personne

  // test

  // création d'un tableau d'objet personne
  $groupe=array(new personne("Paul","Langevin",48), new personne("Sylvie","Lefur",70));

  // identité de ces personnes
  for ($i=0;$i<count($groupe);$i++)
    print "groupe[$i]=".$groupe[$i]->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
groupe[0]=[Paul,Langevin,48]
groupe[1]=[Sylvie,Lefur,70]
```

3.8 Création d'une classe dérivée de la classe personne, polymorphisme

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;

    // constructeur
    function personne($prénom,$nom,$âge){
      $this->prénom=$prénom;
      $this->nom=$nom;
      $this->âge=$âge;
    }//personne

    // méthode
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité
  }//personne

  // une classe dérivée de personne
  class enseignant extends personne{

    // attributs
    var $discipline;   // discipline enseignée

    // constructeur
    function enseignant($prénom,$nom,$âge,$discipline){
```

```
      // attributs parent
      $this->personne($prénom,$nom,$âge);
      // autres attributs
      $this->discipline=$discipline;
    }

    // surcharge de la fonction identité de la classe parente
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge,$this->discipline]";
    }//identité
  }// classe enseignant

  // test

  // création d'un tableau d'objets personne et dérivés
  $groupe=array(new enseignant("Paul","Langevin",48,"anglais"), new
personne("Sylvie","Lefur",70));

  // identité de ces personnes (polymorphisme)
  for ($i=0;$i<count($groupe);$i++)
    print "groupe[$i]=".$groupe[$i]->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
groupe[0]=[Paul,Langevin,48,anglais]
groupe[1]=[Sylvie,Lefur,70]
```

3.9 Création d'une seconde classe dérivée de la classe personne, polymorphisme

```
<?
  class personne{
    // attributs de la classe
    var $prénom;
    var $nom;
    var $âge;

    // constructeur
    function personne($prénom,$nom,$âge){
      $this->prénom=$prénom;
      $this->nom=$nom;
      $this->âge=$âge;
    }//personne

    // méthode
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge]";
    }//identité
  }//personne

  // une classe dérivée de personne
  class enseignant extends personne{

    // attributs
    var $discipline;   // discipline enseignée

    // constructeur
    function enseignant($prénom,$nom,$âge,$discipline){
      // attributs parent
      $this->personne($prénom,$nom,$âge);
      // autres attributs
      $this->discipline=$discipline;
    }

    // surcharge fonction identité
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge,$this->discipline]";
    }//identité
  }// classe enseignant

  // une autre classe dérivée de personne
  class étudiant extends personne{
```

```
    // attributs
    var $formation;   // formation suivie par l'étudiant

    // constructeur
    function étudiant($prénom,$nom,$âge,$formation){
      // attributs parent
      $this->personne($prénom,$nom,$âge);
      // autres attributs
      $this->formation=$formation;
    }

    // surcharge fonction identité
    function identité(){
      return "[$this->prénom,$this->nom,$this->âge,$this->formation]";
    }//identité
  }// classe enseignant

  // test

  // création d'un tableau d'objets personne et dérivés
  $groupe=array(new enseignant("Paul","Langevin",48,"anglais"), new
personne("Sylvie","Lefur",70), new étudiant("Steve","Boer",23,"iup2 qualité"));

  // identité de ces personnes (polymorphisme)
  for ($i=0;$i<count($groupe);$i++)
    print "groupe[$i]=".$groupe[$i]->identité()."\n";
  // fin
  exit;
?>
```

Résultats :

```
groupe[0]=[Paul,Langevin,48,anglais]
groupe[1]=[Sylvie,Lefur,70]
groupe[2]=[Steve,Boer,23,iup2 qualité]
```

4 Exercice d'application - IMPOTS avec objets

On reprend l'exercice déjà étudié dans les pages précédentes.

```
<?
  // définition d'une classe dataImpôts
  class impôts{
    // attributs : les 3 tableaux de données
    var $limites;
    var $coeffR;
    var $coeffN;
    var $erreur;
    var $nbLimites; // nbre d'éléments dans tableau $limites

    // constructeur
    function impôts($IMPOTS){
      // $IMPOTS : le nom du fichier contenant les données des tables $limites, $coeffR,
$coeffN
      // le fichier $IMPOTS existe-t-il ?
      if (! file_exists($IMPOTS)){
        $this->erreur="Le fichier $IMPOTS n'existe pas";
        return;
      }//if
      // lecture en bloc du fichier
      $tables=file($IMPOTS);
      if (! $tables){
       $this->erreur="Erreur lors de l'exploitation du fichier $IMPOTS";
       return;
      }//if
      // création des 3 tables - on suppose que les lignes sont syntaxiquement correctes
      $u=new utilitaires(); // pour disposer de fonctions utilitaires
      $this->limites=explode(":",$u->cutNewLineChar($tables[0]));
      $this->coeffR=explode(":",$u->cutNewLineChar($tables[1]));
      $this->coeffN=explode(":",$u->cutNewLineChar($tables[2]));
      $this->nbLimites=count($this->limites);
      // pas d'erreur
      $this->erreur="";
    }//constructeur impôts

    // ------------------------------------------------------------------------
    function calculer($marié,$enfants,$salaire){
      // $marié : oui, non
      // $enfants : nombre d'enfants
      // $salaire : salaire annuel

      // l'objet est-il dans un état correct ?
      if ($this->erreur) return -1;

      // nombre de parts
      $marié=strToLower($marié);
      if($marié=="oui") $nbParts=$enfants/2+2;
        else $nbParts=$enfants/2+1;
      // une 1/2 part de plus si au moins 3 enfants
      if($enfants>=3) $nbParts+="0,5";
      // revenu imposable
      $revenuImposable="0,72"*$salaire;
      // quotient familial
      $quotient=$revenuImposable/$nbParts;
      // est mis à la fin du tableau limites pour arrêter la boucle qui suit
      $this->limites[$this->nbLimites]=$quotient;
      // calcul de l'impôt
      $i=0;
      while($quotient>$this->limites[$i]) $i++;
      // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
      // ne peut déborder du tableau $limites
      // maintenant on peut calculer l'impôt
      return floor($revenuImposable*$this->coeffR[$i]-$nbParts*$this->coeffN[$i]);
    }//calculImpots
  }//classe impôts

  // une classe de fonctions utilitaires
  class utilitaires{
    function cutNewLinechar($ligne){
```

```
    // on supprime la marque de fin de ligne de $ligne si elle existe
      $L=strlen($ligne);  // longueur ligne
      while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
        $ligne=substr($ligne,0,$L-1);
        $L--;
      }//while
      // fin
      return($ligne);
    }//cutNewLineChar
  }//classe utilitaires


  // test ----------------------------------------------------
  // définition des constantes
  $DATA="data.txt";
  $RESULTATS="résultats.txt";
  $IMPOTS="impots.txt";
  // les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
  // à raison d'une ligne par tableau sous la forme
  // "val1":"val2":"val3",...
  // on crée un objet impôts
  $I=new impôts($IMPOTS);
  // y-a-t-il eu une erreur ?
  if($I->erreur){
    print "$I->erreur\n";
    exit;
  }//if

  // on crée un objet utilitaires
  $u=new utilitaires();

  // lecture des données utilisateur
  $data=@fopen($DATA,"r");
  if(! $data){
    print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
    exit;
  }

  // ouverture fichier des résultats
  $résultats=@fopen($RESULTATS,"w");
  if(! $résultats){
    print "Impossible de créer le fichier des résultats [$RESULTATS]\n";
    exit;
  }

  // on exploite la ligne courante du fichier des données
  while($ligne=fgets($data,100)){
    // on enlève l'éventuelle marque de fin de ligne
    $ligne=$u->cutNewLineChar($ligne);
    // on récupère les 3 champs marié:enfants:salaire qui forment $ligne
    list($marié,$enfants,$salaire)=explode(",",$ligne);
    // on calcule l'impôt
    $impôt=$I->calculer($marié,$enfants,$salaire);
    // on inscrit le résultat
    fputs($résultats,"$marié:$enfants:$salaire:$impôt\n");
    // donnée suivante
  }// while

  // on ferme les fichiers
  fclose($data);
  fclose($résultats);

  // fin
  exit;
?>
```

Résultats : ceux déjà obtenus dans les versions avec tableaux et fichiers.

5 PHP-MySql

5.1 Connexion à une base MySQL - 1

```
<?
  // connexion à une base MySql

  // l'identité de l'utilisateur est (admpersonnes,nobody)
  $ID="admpersonnes";
  $PWD="nobody";
  $HOTE="localhost";
  $connexion=@mysql_connect($HOTE,$ID,$PWD);

  // a-t-on réussi
  if ($connexion){
    print "Connexion à MySQL réussie sous l'identité ($HOTE,$ID,$PWD)\n";
    // on ferme la connexion
    if (mysql_close($connexion))
      print "Fermeture connexion MySql réussie";
    else
      print "Echec de la fermeture de la connexion MySQL (" . mysql_error() .")\n";
  }
  else
    print "Echec de la connexion à MySQL sous l'identité ($HOTE,$ID,$PWD)\n";
?>
```

Résultats :

```
Connexion à MySQL réussie sous l'identité (localhost,admpersonnes,nobody)
Fermeture connexion MySql réussie
```

5.2 Connexion à une base MySQL - 2

```
<?
  // connexion à la base MySql

  // l'identité de l'utilisateur
  $ID="admpersonnes";
  $PWD="nobody";
  $HOTE="localhost";
  testConnexion($HOTE,$ID,$PWD);
  testConnexion($HOTE,"xx","xx");
  exit;

  function testConnexion($mysql,$login,$pwd){
    // connecte puis déconnecte ($login,$pwd) à la base mysql $mysql
    // connexion
    list($erreur,$connexion)=connecte($mysql,$login,$pwd);
    // a-t-on réussi
    if (! $erreur){
      print "Connexion à MySQL réussie sous l'identité ($mysql,$login,$pwd)\n";
      // on ferme la connexion
      if (! ($erreur=ferme($connexion)))
        print "Fermeture connexion MySql réussie\n";
      else
        print "Echec de la fermeture de la connexion MySQL ($erreur)\n";
    }
    else
      print "Echec de la connexion à MySQL sous l'identité ($mysql,$login,$pwd) : $erreur\n";
  }//testConnexion

  // ----------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte
```

```
  // ------------------------------------------------------------------------------
  function ferme($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

?>
```

Résultats :

```
Connexion à MySQL réussie sous l'identité (localhost,admpersonnes,nobody)
Fermeture connexion MySql réussie
Echec de la connexion à MySQL sous l'identité (localhost,xx,xx) : MySQL Connection Failed:
Access denied for user: 'xx@localhost' (Using password: YES)
```

5.3 Création d'une table MySQL

```
<?
  // connexion à la base MySql

  // l'identité de l'utilisateur
  $ID="admpersonnes";
  $PWD="nobody";
  $HOTE="localhost";
  // identité de la base
  $BASE="personnes";

  // connexion
  list($erreur,$connexion)=connecte($HOTE,$ID,$PWD);
  if ($erreur){
    print "Erreur lors de la connexion à MySql sous l'identité ($HOTE,$ID,$PWD) : $erreur\n";
    exit;
  }

  // suppression de la table personnes si elle existe
  // si elle n'existe pas une erreur se produira
  // elle est ignorée
  $requête="drop table personnes";
  exécuteRequête($connexion,$BASE,$requête);
  // création de la table personnes
  $requête="create table personnes (prenom varchar(30) NOT NULL, nom varchar(30) NOT NULL, age
integer NOT NULL, primary key(nom,prenom))";
  list($erreur,$res)=exécuteRequête($connexion,$BASE,$requête);
  //y-a-t-il eu une erreur ?
  if($erreur)
    print "$requête : Erreur ($erreur)\n";
  else
    print "$requête: Exécution réussie\n";
  // on se déconnecte et on quitte
  déconnecte($connexion);
  exit;

  // ------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
```

```
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête
?>
```

Résultats écran :

```
create table personnes (prenom varchar(30) NOT NULL, nom varchar(30) NOT NULL, age integer NOT
NULL, primary key(nom,prenom)): Exécution réussie
```

Vérification avec un client MySql :

```
C:\mysql\bin>mysqlc -u admpersonnes -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql> use personnes
Reading table information for completion of table and column names

Database changed
mysql> show tables;
+----------------------+
| Tables in personnes |
+----------------------+
| personnes           |
+----------------------+
1 row in set (0.06 sec)

mysql> describe personnes;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| prenom | varchar(30) |      | PRI |         |       |
| nom    | varchar(30) |      | PRI |         |       |
| age    | int(11)     |      |     | 0       |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

5.4 Remplissage de la table personnes

```
<?
  // connexion à la base MySql

  // l'identité de l'utilisateur
  $ID="admpersonnes";
  $PWD="nobody";
  $HOTE="localhost";
  // identité de la base
  $BASE="personnes";
  // identité du fichier texte des commandes SQL à exécuter
  $TEXTE="création.txt";

  // connexion
  list($erreur,$connexion)=connecte($HOTE,$ID,$PWD);
  if ($erreur){
    print "Erreur lors de la connexion à MySql sous l'identité ($HOTE,$ID,$PWD) : $erreur\n";
    exit;
  }

  // création et remplissage de la table
  $erreurs=exécuterCommandes($connexion,$BASE,$TEXTE,1,0);
  //affichage nombre d'erreurs
    print "il y a eu $erreurs[0] erreurs\n";
    for($i=1;$i<count($erreurs);$i++)
      print "$erreurs[$i]\n";
```

```
  // on se déconnecte et on quitte
  déconnecte($connexion);
  exit;

  function exécuterCommandes($connexion,$BASE,$SQL,$suivi=0,$arrêt=1){
    // utilise la connexion mysql $connexion
    // exécute dans la base $BASE les commandes SQL contenues dans le fichier texte $SQL
    // ce fichier est un fichier de commandes SQL à exécuter à raison d'une par ligne
    // si $suivi=1 alors chaque exécution d'un ordre SQL fait l'objet d'un affichage indiquant
sa réussite ou son échec
    // si $arrêt=1, la fonction s'arrête sur la 1ère erreur rencontrée sinon elle exécute ttes
les commandes sql
    // la fonction rend un tableau (nb d'erreurs, erreur1, erreur2, ...)

    // on vérifie la présence du fichier $SQL
    if (! file_exists($SQL)) return array(1,"Le fichier $SQL n'existe pas");

    // exécution des requêtes SQL contenues dans $TEXTE
    // on les met dans un tableau
    $requêtes=file($SQL);
    // on les exécute - au départ pas d'erreurs
    $erreurs=array(0);
    for ($i=0;$i<count($requêtes);$i++){
      // on enlève de la requête les éventuels espaces non significatifs ainsi que la marque
de fin de ligne
      if (preg_match("/^\s*(.*?)\s*$/",$requêtes[$i],$champs))
        $requêtes[$i]=$champs[1];
      //a-t-on une requête vide
      if(preg_match("/^\s*$/",$requêtes[$i])) continue;
      // exécution de la requête $i
      list($erreur,$res)=exécuteRequête($connexion,$BASE,$requêtes[$i]);
      //y-a-t-il eu une erreur ?
      if($erreur){
        // une erreur de plus
        $erreurs[0]++;
        // msg d'erreur
        $msg="$requêtes[$i] : Erreur ($erreur)\n";
        $erreurs[]=$msg;
        // suivi écran ou non ?
        if ($suivi) print "$msg\n";
        // on s'arrête ?
        if ($arrêt) return $erreurs;
      } else
          if ($suivi) print "$requêtes[$i] : Exécution réussie\n";
    }//for
    // retour
    return $erreurs;
  }//créerTable

  // ------------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête
?>
```

Le fichier *création.txt* :

```
drop table personnes
create table personnes (prenom varchar(30) not null, nom varchar(30) not null, age integer not
null, primary key (nom,prenom))
insert into personnes values('Paul','Langevin',48)
insert into personnes values ('Sylvie','Lefur',70)
insert into personnes values ('Pierre','Nicazou',35)
insert into personnes values ('Geraldine','Colou',26)
insert into personnes values ('Paulette','Girond',56)
```

Les résultats écran :

```
drop table personnes : Exécution réussie
create table personnes (prenom varchar(30) not null, nom varchar(30) not null, age integer not
null, primary key (nom,prenom)) : Exécution réussie
insert into personnes values('Paul','Langevin',48) : Exécution réussie
insert into personnes values ('Sylvie','Lefur',70) : Exécution réussie
insert into personnes values ('Pierre','Nicazou',35) : Exécution réussie
insert into personnes values ('Geraldine','Colou',26) : Exécution réussie
insert into personnes values ('Paulette','Girond',56) : Exécution réussie
il y a eu 0 erreurs
```

Vérification avec MySQL :

```
mysql> select * from personnes;
+-----------+----------+-----+
| prenom    | nom      | age |
+-----------+----------+-----+
| Paul      | Langevin |  48 |
| Sylvie    | Lefur    |  70 |
| Pierre    | Nicazou  |  35 |
| Geraldine | Colou    |  26 |
| Paulette  | Girond   |  56 |
+-----------+----------+-----+
5 rows in set (0.07 sec)
```

5.5 Exécution de requêtes SQL quelconques

```
<?
  // connexion à la base MySql

  // l'identité de l'utilisateur
  $ID="admpersonnes";
  $PWD="nobody";
  $HOTE="localhost";
  // identité de la base
  $BASE="personnes";
  // identité du fichier texte des commandes SQL à exécuter
  $TEXTE="sql.txt";

  // connexion
  list($erreur,$connexion)=connecte($HOTE,$ID,$PWD);
  if ($erreur){
    print "Erreur lors de la connexion à MySql sous l'identité ($HOTE,$ID,$PWD) : $erreur\n";
    exit;
  }

  // exécution des commandes sql
  $erreurs=exécuterCommandes($connexion,$BASE,$TEXTE,1,0);
  //affichage nombre d'erreurs
    print "il y a eu $erreurs[0] erreur(s)\n";
    for($i=1;$i<count($erreurs);$i++)
      print "$erreurs[$i]\n";

  // on se déconnecte et on quitte
  déconnecte($connexion);
  exit;

  // ------------------------------------------------------------------------------
  function afficherInfos($connexion,$résultat){
```

```
    // affiche le résultat $résultat d'une requête sql
    // s'agissait-il d'un select ?
    if ($nbColonnes=@mysql_num_fields($résultat)){
      // il y a des champs - donc c'est un select
      // on affiche les noms des champs
      $titre="";
      for ($i=0;$i<$nbColonnes;$i++)
        $titre.=mysql_field_name($résultat,$i).",";
      // on enlève le dernier caractère ,
      $titre=substr($titre,0,strlen($titre)-1);
      // on affiche la liste des champs
      print "$titre\n";
      // ligne séparatrice
      for($i=0;$i<strlen($titre);$i++)
        $séparateurs.="-";
      print "$séparateurs\n";
      // données
      while($colonnes=mysql_fetch_row($résultat))
        print join(",",$colonnes)."\n";
    }else
      // il n'y a pas de champs - ce n'était pas un select
      print mysql_affected_rows($connexion)." lignes(s) ont été modifiées\n";
  }//afficherInfos

  function exécuterCommandes($connexion,$BASE,$SQL,$suivi=0,$arrêt=1){
    // utilise la connexion mysql $connexion
    // exécute dans la base $BASE les commandes SQL contenues dans le fichier texte $SQL
    // ce fichier est un fichier de commandes SQL à exécuter à raison d'une par ligne
    // si $suivi=1 alors chaque exécution d'un ordre SQL fait l'objet d'un affichage indiquant
sa réussite ou son échec
    // si $arrêt=1, la fonction s'arrête sur la 1ère erreur rencontrée sinon elle exécute ttes
les commandes sql
    // la fonction rend un tableau (nb d'erreurs, erreur1, erreur2, ...)

    // on vérifie la présence du fichier $SQL
    if (! file_exists($SQL)) return array(1,"Le fichier $SQL n'existe pas");

    // exécution des requêtes SQL contenues dans $TEXTE
    // on les met dans un tableau
    $requêtes=file($SQL);
    // on les exécute - au départ pas d'erreurs
    $erreurs=array(0);
    for ($i=0;$i<count($requêtes);$i++){
      // on enlève de la requête les éventuels espaces non significatifs ainsi que la marque
de fin de ligne
      if (preg_match("/^\s*(.*?)\s*$/",$requêtes[$i],$champs))
        $requêtes[$i]=$champs[1];
      //a-t-on une requête vide
      if(preg_match("/^\s*$/",$requêtes[$i])) continue;
      // exécution de la requête $i
      list($erreur,$res)=exécuteRequête($connexion,$BASE,$requêtes[$i]);
      //y-a-t-il eu une erreur ?
      if($erreur){
        // une erreur de plus
        $erreurs[0]++;
        // msg d'erreur
        $msg="$requêtes[$i] : Erreur ($erreur) lors de l'exécution de la requête $i\n";
        $erreurs[]=$msg;
        // suivi écran ou non ?
        if ($suivi) print "$msg\n";
        // on s'arrête ?
        if ($arrêt) return $erreurs;
      } else
          if ($suivi){
            print "$requêtes[$i] : Exécution réussie de la requête $i\n";
            // infos sur le résultat de la requête exécutée
            afficherInfos($connexion,$res);
          }//if
    }//for
    // retour
    return $erreurs;
  }//créerTable

  // ------------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
```

```
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête
?>
```

Le fichier des requêtes exécutées :

```
select * from personnes
select nom,prenom from personnes order by nom asc, prenom desc
select * from personnes where age between 20 and 40 order by age desc, nom asc, prenom asc
insert into personnes values('Josette','Bruneau',46)
update personnes set age=47 where nom='Bruneau'
select * from personnes where nom='Bruneau'
delete from personnes where nom='Bruneau'
select * from personnes where nom='Bruneau'
xselect * from personnes where nom='Bruneau'
```

Les résultats écran :

```
select * from personnes : Exécution réussie de la requête 0
prenom,nom,age
--------------
Paul,Langevin,48
Sylvie,Lefur,70
Pierre,Nicazou,35
Geraldine,Colou,26
Paulette,Girond,56
select nom,prenom from personnes order by nom asc, prenom desc : Exécution réussie de la
requête 1
nom,prenom
----------
Colou,Geraldine
Girond,Paulette
Langevin,Paul
Lefur,Sylvie
Nicazou,Pierre
select * from personnes where age between 20 and 40 order by age desc, nom asc, prenom asc :
Exécution réussie de la requête 2
prenom,nom,age
--------------
Pierre,Nicazou,35
Geraldine,Colou,26
insert into personnes values('Josette','Bruneau',46) : Exécution réussie de la requête 3
1 lignes(s) ont été modifiées
update personnes set age=47 where nom='Bruneau' : Exécution réussie de la requête 4
1 lignes(s) ont été modifiées
select * from personnes where nom='Bruneau' : Exécution réussie de la requête 5
prenom,nom,age
--------------
Josette,Bruneau,47
delete from personnes where nom='Bruneau' : Exécution réussie de la requête 6
1 lignes(s) ont été modifiées
select * from personnes where nom='Bruneau' : Exécution réussie de la requête 7
prenom,nom,age
--------------
```

```
xselect * from personnes where nom='Bruneau' : Erreur (You have an error in your SQL syntax
near 'xselect * from personnes where nom='Bruneau'' at line 1) lors de l'exécution de la
requête 8

il y a eu 1 erreur(s)
xselect * from personnes where nom='Bruneau' : Erreur (You have an error in your SQL syntax
near 'xselect * from personnes where nom='Bruneau'' at line 1) lors de l'exécution de la
requête 8
```

Vérification MySQL :

```
mysql> select * from personnes;
+-----------+----------+-----+
| prenom    | nom      | age |
+-----------+----------+-----+
| Paul      | Langevin |  48 |
| Sylvie    | Lefur    |  70 |
| Pierre    | Nicazou  |  35 |
| Geraldine | Colou    |  26 |
| Paulette  | Girond   |  56 |
+-----------+----------+-----+
5 rows in set (0.03 sec)
```

6 Exercice IMPOTS avec MySQL

6.1 Transfert d'un fichier texte dans une table MySQL

```
<?
  // transfère le fichier texte des données nécessaires au calcul des impôts
  // dans une table mysql

  // les données
  $IMPOTS="impots.txt";
  $BASE="impots";
  $TABLE="impots";
  $USER="admimpots";
  $PWD="mdpimpots";
  $HOTE="localhost";

  // les données nécessaires au calcul de l'impôt ont été placées dans le fichier IMPOTS
  // à raison d'une ligne par tableau sous la forme
  // val1:val2:val3,...
  list($erreur,$limites,$coeffR,$coeffN)=getTables($IMPOTS);
  // y-a-t-il eu une erreur ?
  if($erreur){
    print "$erreur\n";
    exit;
  }//if
  // on transfère ces tableaux dans une table mysql
  $erreur=copyToMysql($limites,$coeffR,$coeffN,$HOTE,$USER,$PWD,$BASE,$TABLE);
  if ($erreur)
    print "$erreur\n";
  else print "Transfert opéré\n";
  // fin
  exit;

  // ------------------------------------------------------------------------
  function copyToMysql($limites,$coeffR,$coeffN,$HOTE,$USER,$PWD,$BASE,$TABLE){
    // copie les 3 tableaux numériques $limites, $coeffR, $coeffN
    // dans la table $TABLE de la base mysql $BASE
    // la base mysql est sur la machine $HOTE
    // l'utilisateur est identifié par $USER et $PWD

    // connexion
    list($erreur,$connexion)=connecte($HOTE,$USER,$PWD);
    if ($erreur)
      return "Erreur lors de la connexion à MySql sous l'identité ($HOTE,$USER,$PWD) :
$erreur\n";
    // suppression de la table
    $requête="drop table $TABLE";
    exécuteRequête($connexion,$BASE,$requête);
    // création de la table
    $requête="create table $TABLE (limites decimal(10,2), coeffR decimal(6,2), coeffN
decimal(10,2))";
    list($erreur,$res)=exécuteRequête($connexion,$BASE,$requête);
    if($erreur) return "$requête : erreur ($erreur)";
    // remplissage
    for($i=0;$i<count($limites);$i++){
      // requête d'insertion
      $requête="insert into $TABLE (limites,coeffR,coeffN) values
($limites[$i],$coeffR[$i],$coeffN[$i])";
      // exécution de la requête
      list($erreur,$res)=exécuteRequête($connexion,$BASE,$requête);
      // retour si erreur
      if($erreur) return "$requête : erreur ($erreur)";
    }//for
    // c'est fini - on se déconnecte
    déconnecte($connexion);
    // retour sans erreur
    return "";
  }//copyToMysql

  // ------------------------------------------------------------------------
  function getTables($IMPOTS){
    // $IMPOTS : le nom du fichier contenant les données des tables $limites, $coeffR, $coeffN
    // le fichier $IMPOTS existe-t-il ?
```

```
    if (! file_exists($IMPOTS)) return array("Le fichier $IMPOTS n'existe pas");
    // lecture en bloc du fichier
    $tables=file($IMPOTS);
    if (! $tables) return array("Erreur lors de l'exploitation du fichier $IMPOTS");
    // création des 3 tables - on suppose que les lignes sont syntaxiquement correctes
    $limites=explode(":",cutNewLineChar($tables[0]));
    $coeffR=explode(":",cutNewLineChar($tables[1]));
    $coeffN=explode(":",cutNewLineChar($tables[2]));
    // fin
    return array("",$limites,$coeffR,$coeffN);
  }//getTables

  // ------------------------------------------------------------------------
  function cutNewLinechar($ligne){
  // on supprime la marque de fin de ligne de $ligne si elle existe
    $L=strlen($ligne);  // longueur ligne
    while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
      $ligne=substr($ligne,0,$L-1);
      $L--;
    }
    // fin
    return($ligne);
  }//cutNewLineChar

 // ------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête
?>
```

Le fichier texte *impots.txt* :

```
12620:13190:15640:24740:31810:39970:48360:55790:92970:127860:151250:172040:195000:0
0:0.05:0.1:0.15:0.2:0.25:0.3:0.35:0.4:0.45:0.5:0.55:0.6:0.65
0:631:1290.5:272.5:3309.5:4900:6898.5:9316.5:12106:16754.5:23147.5:30710:39312:49062
```

Les résultats écran :

```
Transfert opéré
```

Vérification avec MySQL :

```
C:\mysql\bin>mysqlc -u admimpots -p
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11 to server version: 3.22.34-shareware-debug

Type 'help' for help.

mysql> use impots
Reading table information for completion of table and column names
```

```
Database changed
mysql> show tables;
+------------------+
| Tables in impots |
+------------------+
| impots           |
+------------------+
1 row in set (0.02 sec)

mysql> select * from impots;
+-----------+--------+----------+
| limites   | coeffR | coeffN   |
+-----------+--------+----------+
|  12620.00 |   0.00 |     0.00 |
|  13190.00 |   0.05 |   631.00 |
|  15640.00 |   0.10 |  1290.50 |
|  24740.00 |   0.15 |   272.50 |
|  31810.00 |   0.20 |  3309.50 |
|  39970.00 |   0.25 |  4900.00 |
|  48360.00 |   0.30 |  6898.50 |
|  55790.00 |   0.35 |  9316.50 |
|  92970.00 |   0.40 | 12106.00 |
| 127860.00 |   0.45 | 16754.50 |
| 151250.00 |   0.50 | 23147.50 |
| 172040.00 |   0.55 | 30710.00 |
| 195000.00 |   0.60 | 39312.00 |
|      0.00 |   0.65 | 49062.00 |
+-----------+--------+----------+
14 rows in set (0.03 sec)
```

6.2 Le programme de calcul de l'impôt

```
<?
  // test ----------------------------------------------------
  // définition des constantes
  $DATA="data.txt";
  $RESULTATS="résultats.txt";
  $TABLE="impots";
  $BASE="impots";
  $USER="admimpots";
  $PWD="mdpimpots";
  // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL $TABLE
  // appartenant à la base $BASE. La table a la structure suivante
  // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
  // les paramètres des personnes imposables (statut marital, nombre d'enfants, salaire
annuel)
  // ont été placés dans le fichier texte $DATA à raison d'une ligne par contribuable
  // les résultats (statut marital, nombre d'enfants, salaire annuel, impôt à payer) sont
placés dans
  // le fichier texte $RESULTATS à raison d'un résultat par ligne

  // on crée un objet impôts
  $I=new impôts($HOTE,$USER,$PWD,$BASE,$TABLE);
  // y-a-t-il eu une erreur ?
  if($I->erreur){
    print "$I->erreur\n";
    exit;
  }//if

  // on crée un objet utilitaires
  $u=new utilitaires();

  // ouverture fichier des données des contribuables
  $data=@fopen($DATA,"r");
  if(! $data){
    print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
    exit;
  }

  // ouverture fichier des résultats
  $résultats=@fopen($RESULTATS,"w");
  if(! $résultats){
    print "Impossible de créer le fichier des résultats [$RESULTATS]\n";
    exit;
```

```
  }

  // on exploite la ligne courante du fichier des données contribuables
  while($ligne=fgets($data,100)){
    // on enlève l'éventuelle marque de fin de ligne
    $ligne=$u->cutNewLineChar($ligne);
    // on récupère les 3 champs marié:enfants:salaire qui forment $ligne
    list($marié,$enfants,$salaire)=explode(",",$ligne);
    // on calcule l'impôt
    $impôt=$I->calculer($marié,$enfants,$salaire);
    // on inscrit le résultat
    fputs($résultats,"$marié:$enfants:$salaire:$impôt\n");
    // donnée suivante
  }// while

  // on ferme les fichiers
  fclose($data);
  fclose($résultats);

  // fin
  exit;

  // ------------------------------------------------------------------------------------
  // définition d'une classe dataImpôts
  class impôts{
    // attributs : les 3 tableaux de données
    var $limites;
    var $coeffR;
    var $coeffN;
    var $erreur;
    var $nbLimites; // nbre d'éléments dans tableau $limites

    // constructeur
    function impôts($HOTE,$USER,$PWD,$BASE,$TABLE){
      // initialise les attributs $limites, $coeffR, $coeffN
      // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL
$TABLE
      // appartenant à la base $BASE. La table a la structure suivante
      // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
      // la connexion à la base mysql de la machine $HOTE se fait sous l'identité ($USER,$PWD)
      // rend une chaîne $erreur indiquant la nature d'une éventuelle erreur
      // vide si pas d'erreur

      // connexion à la base mysql
        list($erreur,$connexion)=connecte($HOTE,$USER,$PWD);
        if ($erreur){
          $this->erreur="Erreur lors de la connexion à MySql sous l'identité
($HOTE,$USER,$PWD) : $erreur\n";
          return;
        }
      // lecture en bloc de la table $TABLE
        $requête="select limites,coeffR,coeffN from $TABLE";
        // exécute la requête $requête sur la base $base de la connexion $connexion
        if (! ($résultat=@mysql_db_query($BASE,$requête,$connexion))){
           $this->erreur="$requête : ".mysql_error();
           return;
        }//if
        // exploitation du résultat de la requête
        while($colonnes=mysql_fetch_row($résultat)){
          // les nombres mysql sont de la forme eee.dd
          // alors que php4/windows veut eee,dd à cause d'un bogue
          for($i=0;$i<count($colonnes);$i++)
            $colonnes[$i]=str_replace(".",",",$colonnes[$i]);
          $this->limites[]=$colonnes[0];
          $this->coeffR[]=$colonnes[1];
          $this->coeffN[]=$colonnes[2];
        }//while
      // déconnexion
      déconnecte($connexion);
      // pas d'erreur
      $this->erreur="";
      // nombre d'élements de limites
      $this->nbLimites=count($this->limites);
    }//constructeur impôts

    // ------------------------------------------------------------------------
    function calculer($marié,$enfants,$salaire){
```

```
      // $marié : oui, non
      // $enfants : nombre d'enfants
      // $salaire : salaire annuel

      // l'objet est-il dans un état correct ?
      if ($this->erreur) return -1;

      // nombre de parts
      $marié=strToLower($marié);
      if($marié=="oui") $nbParts=$enfants/2+2;
        else $nbParts=$enfants/2+1;
      // une 1/2 part de plus si au moins 3 enfants
      if($enfants>=3) $nbParts+="0,5";
      // revenu imposable
      $revenuImposable="0,72"*$salaire;
      // quotient familial
      $quotient=$revenuImposable/$nbParts;
      // est mis à la fin du tableau limites pour arrêter la boucle qui suit
      $N=$this->nbLimites;
      $this->limites[$N-1]=$quotient;
      // calcul de l'impôt
      $i=0;
      while($i<$this->nbLimites and $quotient>$this->limites[$i]) $i++;
      // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
      // ne peut déborder du tableau $limites
      // maintenant on peut calculer l'impôt
      return floor($revenuImposable*$this->coeffR[$i]-$nbParts*$this->coeffN[$i]);
    }//calculImpots
  }//classe impôts

  // une classe de fonctions utilitaires
  class utilitaires{
    function cutNewLinechar($ligne){
    // on supprime la marque de fin de ligne de $ligne si elle existe
      $L=strlen($ligne);  // longueur ligne
      while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
        $ligne=substr($ligne,0,$L-1);
        $L--;
      }//while
      // fin
      return($ligne);
    }//cutNewLineChar
  }//classe utilitaires

 // ------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête

?>
```

Résultats : les mêmes qu'avec les versions précédentes.

7 Les fonctions réseau de PHP

7.1 Obtenir le nom ou l'adresse IP d'une machine de l'Internet

```
<?
  // fonctions nom Machine <--> adresse IP machine

  // constantes
  $HOTES=array("istia.univ-angers.fr","www.univ-angers.fr","www.ibm.com","localhost","","xx");

  // adresses IP des machines de $HOTES
  for($i=0;$i<count($HOTES);$i++)
    getIPandName($HOTES[$i]);
  // fin
  exit;

  //-----------------------------------------------
  function getIPandName($nomMachine){
    //$nomMachine : nom de la machine dont on veut l'adresse IP
    // nomMachine-->adresse IP
    $ip=gethostbyname($nomMachine);
    print "ip[$nomMachine]=$ip\n";
    // adresse IP --> nomMachine
    $name=gethostbyaddr($ip);
    print "name[$ip]=$name\n";
  }//getIP
?>
```

Résultats :

```
ip[istia.univ-angers.fr]=193.49.146.171
name[193.49.146.171]=istia.univ-angers.fr
ip[www.univ-angers.fr]=193.49.144.131
name[193.49.144.131]=web-serv.univ-angers.fr
ip[www.ibm.com]=129.42.19.99
name[129.42.19.99]=www.ibm.com
ip[localhost]=127.0.0.1
name[127.0.0.1]=LOCALHOST
ip[]=193.248.226.211
name[193.248.226.211]=Mix-LeMans-210-2-211.abo.wanadoo.fr
ip[xx]=xx
name[xx]=xx
```

7.2 Un client web

```
<?
  // otenir le texte HTML d'URL

  // liste de sites web
  $SITES=array("istia.univ-angers.fr","www.univ-angers.fr","www.ibm.com","xx");

  // lecture des pages index des sites du tableau $SITES
  for($i=0;$i<count($SITES);$i++){
    // lecture page index du site $SITES[$i]
    $résultat=getIndex($SITES[$i]);
    // affichage résultat
    print "$résultat\n";
  }//for
  // fin
  exit;

  //-------------------------------------------------------------------------
  function getIndex($site){
    // lit l'URL $site/ et la stocke dans le fichier $site.html

    // création du fichier $site.html
    $html=fopen("$site.html","w");
    if (! $html)
      return "Erreur lors de la création du fichier $site.html";
```

```php
    // ouverture d'une connexion sur le port 80 de $site
    $connexion=fsockopen($site,80,&$errno,&$erreur);
    // retour si erreur
    if(! $connexion)
      return "Echec de la connexion au site ($site,80) : $erreur";

    // $connexion représente un flux de communication bidirectionnel
    // entre le client (ce programme) et le serveur web contacté
    // ce canal est utilisé pour les échanges de commandes et d'informations
    // le protocole de dialogue est HTTP

    // le client envoie la commande get pour demander l'URL /
    // syntaxe get URL HTTP/1.0
    // les entêtes (headers) du protocole HTTP doivent se terminer par une ligne vide
    fputs($connexion, "GET / HTTP/1.0\n\n");

    // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
    // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
    // jusqu'à la fermeture du canal
    while($ligne=fgets($connexion,1000))
      fputs($html,$ligne);

    // le client ferme la connexion à son tour
    fclose($connexion);
    // fermeture du fichier $html
    fclose($html);
    // retour
    return "Transfert réussi de la page index du site $site";
  }//getIndex

?>
```

Résultats : le fichier reçu pour le site istia.univ-angers.fr :

```
HTTP/1.1 200 OK
Date: Fri, 03 Aug 2001 15:35:42 GMT
Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) PHP/3.0.15 mod_perl/1.21
Last-Modified: Thu, 26 Apr 2001 13:59:42 GMT
ETag: "23432-2a90-3ae829ce"
Accept-Ranges: bytes
Content-Length: 10896
Connection: close
Content-Type: text/html

<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
<title>Bienvenue a l'ISTIA - Universite d'Angers - 2</title>
</head>

<body background="/Fonds/STANDARD">

<table border="0">
    <tr>
        <td align="center" valign="top" width="192" height="102"><a
        href="http://www.univ-angers.fr"><img
        src="http://www.istia.univ-angers.fr/Icones/univ01.gif"
        alt="Logo Université" border="0" width="80" height="95"></a><br>
        <a href="http://www.univ-angers.fr"><font size="2"
        face="Verdana">Université d'Angers</font></a><br>
        <a href="http://www.istia.univ-angers.fr/"><font size="2"
        face="Verdana">ISTIA</font></a></td>
        <td align="center" width="539" height="102"><font
        color="#8080FF" size="5" face="Verdana">Institut des
        Sciences et Techniques</font><font color="#8080FF"
        size="1" face="Verdana"><br>
        </font><font color="#8080FF" size="5" face="Verdana">de l'Ingénieur
        d'Angers</font><br>
        <font size="2" face="Verdana">62, avenue Notre Dame du
        Lac - 49000 Angers</font><br>
        <font size="2" face="Verdana">Tél : 02.41.36.57.57 - Fax
```

```
        : 02.41.36.57.58</font></td>
    </tr>
...
</table>

<p><font size="2" face="Verdana">Responsable du document : </font><a
href="mailto:alain.godon@univ-angers.fr"><font size="2"
face="Verdana">Alain Godon</font></a><font size="2"
face="Verdana"> - Dernière mise à jour le <b>23 novembre 2000</b></font></p>
</body>
</html>
```

7.3 Un client smtp

```
<?
  // client SMTP (SendMail Transfer Protocol) permettant d'envoyer un message
  // les infos sont prises dans un fichier $INFOS contenant les lignes suivantes
  // ligne 1 : smtp, expéditeur, destinataire
  // lignes suivantes : le texte du message

  // expéditeur:email expéditeur
  // destinataire: email destinataire
  // smtp: nom du serveur smtp à utiliser


  // protocole de communication SMTP client-serveur
  // -> client se connecte sur le port 25 du serveur smtp
  // <- serveur lui envoie un message de bienvenue
  // -> client envoie la commande EHLO: nom de sa machine
  // <- serveur répond OK ou non
  // -> client envoie la commande mail from: <expéditeur>
  // <- serveur répond OK ou non
  // -> client envoie la commande rcpt to: <destinataire>
  // <- serveur répond OK ou non
  // -> client envoie la commande data
  // <- serveur répond OK ou non
  // -> client envoie ttes les lignes de son message et termine avec une ligne contenant le
seul caractère .
  // <- serveur répond OK ou non
  // -> client envoie la commande quit
  // <- serveur répond OK ou non

  // les réponses du serveur ont la forme xxx texte où xxx est un nombre à 3 chiffres. Tout
nombre xxx >=500
  // signale une erreur. La réponse peut comporter plusieurs lignes commençant toutes par xxx-
sauf la dernière
  // de la forme xxx(espace)

  // les lignes de texte échangées doivent se terminer par les caractères RC(#13) et LF(#10)

  // données
  $INFOS="infos.txt"; // les paramètres de l'envoi du courrier

  // on récupère les paramètres du courrier
  list($erreur,$smtpServer,$expéditeur,$destinataire,$message)=getInfos($INFOS);
  // erreur ?
  if ($erreur){
    print "$erreur\n";
    exit;
  }
  print "Envoi du message [$smtpServer,$expéditeur,$destinataire]\n";

  // envoi du courrier en mode verbeux
  $résultat=sendmail($smtpServer,$expéditeur,$destinataire,$message,1);
  print "Résultat de l'envoi : $résultat\n";
  // fin
  exit;

  //--------------------------------------------------------------------
  function getInfos($fichier){
    // rend les informations ($smtp,$expéditeur,$destinataire,$message) prises dans le fichier
texte $fichier
    // ligne 1 : smtp, expéditeur, destinataire
    // lignes suivantes : le texte du message
```

```
    // ouverture de $fichier
    $infos=@fopen($fichier,"r");
    // le fichier $fichier existe-t-il
    if (! $infos)
      return array("Le fichier $fichier n'a pu être ouvert en lecture");
    // lecture de la 1ère ligne
    $ligne=fgets($infos,1000);
    // suppression de la marque de fin de ligne
    $ligne=cutNewLineChar($ligne);
    // récupération des champs smtp,expéditeur,destinataire
    $champs=explode(",",$ligne);
    // a-t-on le bon nombre de champs ?
    if (count($champs)!=3)
      return "La ligne 1 du fichier $fichier (serveur smtp, expéditeur, destinataire) a un
nombre de champs incorrect";
    // "traitement" des informations récupérées
    for($i=0;$i<count($champs);$i++)
      $champs[$i]=trim($champs[$i]);
    // récupération des champs
    list($smtpServer,$expéditeur,$destinataire)=$champs;
    // lecture message
    $message="";
    while($ligne=fgets($infos,1000))
      $message.=$ligne;
    fclose($infos);
    // retour
    return array("",$smtpServer,$expéditeur,$destinataire,$message);
  }//getInfos

  //-----------------------------------------------------------------------
  function sendmail($smtpServer,$expéditeur,$destinataire,$message,$verbose){
    // envoie $message au serveur smtp $smtpserver de la part de $expéditeur
    // pour $destinataire. Si $verbose=1, fait un suivi des échanges client-serveur

    // on récupère le nom du client
    $client=gethostbyaddr(gethostbyname(""));

    // ouverture d'une connexion sur le port 25 de $smtpServer
    $connexion=fsockopen($smtpServer,25,&$errno,&$erreur);
    // retour si erreur
    if(! $connexion)
      return "Echec de la connexion au site ($smtpServer,25) : $erreur";

    // $connexion représente un flux de communication bidirectionnel
    // entre le client (ce programme) et le serveur smtp contacté
    // ce canal est utilisé pour les échanges de commandes et d'informations

    // après la connexion le serveur envoie un message de bienvenue qu'on lit
    $erreur=sendCommand($connexion,"",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // cmde ehlo:
    $erreur=sendCommand($connexion,"EHLO $client",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // cmde mail from:
    $erreur=sendCommand($connexion,"MAIL FROM: <$expéditeur>",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // cmde rcpt to:
    $erreur=sendCommand($connexion,"RCPT TO: <$destinataire>",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // cmde data
    $erreur=sendCommand($connexion,"DATA",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // envoi message
    $erreur=sendCommand($connexion,"$message\n.\n",$verbose,0);
    if($erreur) {fclose($connexion);return $erreur;}
    // cmde quit
    $erreur=sendCommand($connexion,"QUIT",$verbose,1);
    if($erreur) {fclose($connexion);return $erreur;}
    // fin
    fclose($connexion);
    return "Message envoyé";
  }//sendmail

  // ----------------------------------------------------------------------
  function sendCommand($connexion,$commande,$verbose,$withRCLF){
    // envoie $commande dans le canal $connexion
    // mode verbeux si $verbose=1
```

```
    // si $withRCLF=1, ajoute la séquence RCLF à échange

    // données
    if($withRCLF) $RCLF="\r\n"; else $RCLF="";
    // envoi cmde si $commande non vide
    if($commande){
      fputs($connexion,"$commande$RCLF");
      // écho éventuel
      if ($verbose) affiche($commande,1);
    }//if
    // lecture réponse
    $réponse=fgets($connexion,1000);
    // écho éventuel
    if($verbose) affiche($réponse,2);
    // récupération code erreur
    $codeErreur=substr($réponse,0,3);
    // dernière ligne de la réponse ?
    while(substr($réponse,3,1)=="-"){
      // lecture réponse
      $réponse=fgets($connexion,1000);
      // écho éventuel
      if($verbose) affiche($réponse,2);
    }//while
    // réponse terminée
    // erreur renvoyée par le serveur ?
    if ($codeErreur >=500)
      return substr($réponse,4);
    // retour sans erreur
    return "";
  }//sendCommand

  // ------------------------------------------------------------------------
  function affiche($échange,$sens){
    // affiche $échange à l'écran
    // si $sens=1 affiche -->$echange
    // si $sens=2 affiche <-- $échange sans les 2 derniers caractères RCLF
    switch($sens){
      case 1:
        print "--> [$échange]\n";
        return;
      case 2:
        $L=strlen($échange);
        print "<-- [".substr($échange,0,$L-2)."]\n";
        return;
    }//switch
  }//affiche

  // ------------------------------------------------------------------------
  function cutNewLinechar($ligne){
  // on supprime la marque de fin de ligne de $ligne si elle existe
    $L=strlen($ligne);  // longueur ligne
    while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
      $ligne=substr($ligne,0,$L-1);
      $L--;
    }
    // fin
    return($ligne);
  }//cutNewLineChar

?>
```

Le fichier *infos.txt* :

```
istia.univ-angers.fr, serge.tahe@univ-angers.fr , serge.tahe@istia.univ-angers.fr
Subject: test

ligne1
ligne2

ligne3
```

Les résultats écran :

```
Envoi du message [istia.univ-angers.fr,serge.tahe@univ-angers.fr,serge.tahe@istia.univ-angers.fr]
<-- [220 istia.univ-angers.fr ESMTP Sendmail 8.9.3/8.9.3; Fri, 3 Aug 2001 17:38:27 +0200]
--> [EHLO Mix-LeMans-210-2-211.abo.wanadoo.fr]
<-- [250-istia.univ-angers.fr Hello Mix-LeMans-210-2-211.abo.wanadoo.fr [193.248.226.211], pleased to meet you]
<-- [250-EXPN]
<-- [250-VERB]
<-- [250-8BITMIME]
<-- [250-SIZE 5000000]
<-- [250-DSN]
<-- [250-ONEX]
<-- [250-ETRN]
<-- [250-XUSR]
<-- [250 HELP]
--> [MAIL FROM: <serge.tahe@univ-angers.fr>]
<-- [250 <serge.tahe@univ-angers.fr>... Sender ok]
--> [RCPT TO: <serge.tahe@istia.univ-angers.fr>]
<-- [250 <serge.tahe@istia.univ-angers.fr>... Recipient ok]
--> [DATA]
<-- [354 Enter mail, end with "." on a line by itself]
--> [Subject: test

ligne1
ligne2

ligne3
.
]
<-- [250 RAA00586 Message accepted for delivery]
--> [QUIT]
<-- [221 istia.univ-angers.fr closing connection]
Résultat de l'envoi : Message envoyé
```

8 Des serveurs en PHP

PHP n'a pas les fonctions réseau permettant d'écrire des serveurs. Il ne possède notamment pas la fonction de création d'une socket d'écoute des demandes clientes. Les programmes PHP pouvant être exécutés par un serveur WEB, c'est ce dernier qui sera à l'écoute des demandes clientes. Du point de vue du client, appeler un service WEB revient à demander l'URL de ce service. Le client peut être écrit avec n'importe quel langage, notamment en PHP. Dans ce dernier cas, on utilise alors les fonctions réseau que nous venons de voir. Il nous faut par ailleurs savoir "converser" avec un service WEB, c'est à dire comprendre le protocole *http* de communication entre un serveur Web et ses clients. C'est le but des programmes qui suivent.

8.1 Application client/ serveur de date/heure

8.1.1 Le serveur

```
<?
  // time : nb de millisecondes depuis 01/01/1970
  // "format affichage date-heure
  // d: jour sur 2 chiffres
  // m: mois sur 2 chiffres
  // y : année sur 2 chiffres
  // H : heure 0,23
  // i : minutes
  // s: secondes
  print date("d/m/y H:i:s",time());
?>
```

8.1.2 Un client

```
<?
  // données
  $HOTE="localhost";
  $PORT=80;
  $urlServeur="/php4/dateheure/dateheure.php";

  // ouverture d'une connexion sur le port 80 de $HOTE
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // erreur ?
  if (! $connexion){
    print "Erreur : $erreur\n";
    exit;
  }

  // on se connecte sur une URL au serveur Web
  // le client envoie la commande get pour demander l'URL /
  // syntaxe get URL HTTP/1.0
  // les entêtes (headers) du protocole HTTP doivent se terminer par une ligne vide
  fputs($connexion, "GET $urlServeur HTTP/1.0\n\n");

  // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
  // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
  // jusqu'à la fermeture du canal
  while($ligne=fgets($connexion,1000)){
    print "$ligne";
  }//while

  // le client ferme la connexion à son tour
  fclose($connexion);

  // fin
  exit;
?>
```

8.1.3 Résultats

```
HTTP/1.1 200 OK
Date: Fri, 31 Aug 2001 14:53:38 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

31/08/01 16:53:39
```

Commentaires

On remarquera le résultat envoyé par le script php

```
31/08/01 16:13:38
```

est précédé des entêtes http envoyés par le serveur Web :

```
HTTP/1.1 200 OK
Date: Fri, 31 Aug 2001 14:53:38 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html
```

Ceux-ci se terminent par une ligne vide. C'est à ce signe que le client peut savoir que les entêtes http sont reminés et qu'arrivent les données envoyées par le script PHP.

8.1.4 Un deuxième client

Nous cherchons ici à récupérer les données envoyées par le script PHP ce que ne faisait pas le script précédent qui se contentait d'afficher ce qu'il recevait sans chercher à récupérer quelque chose de précis. Ici, les données qui nous intéressent vont être récupérées au moyen d'une expression régulière.

```
<?
  // récupérer les informations envoyées par un serveur web

  // données
  $HOTE="localhost";
  $PORT=80;
  $urlServeur="/php4/dateheure/dateheure.php";

  // ouverture d'une connexion sur le port 80 de $HOTE
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // erreur ?
  if (! $connexion){
    print "Erreur : $erreur\n";
    exit;
  }

  // on se connecte sur une URL au serveur Web
  // le client envoie la commande get pour demander l'URL /
  // syntaxe get URL HTTP/1.0
  // les entêtes (headers) du protocole HTTP doivent se terminer par une ligne vide
  fputs($connexion, "GET $urlServeur HTTP/1.0\n\n");

  // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
  // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
  // jusqu'à trouver la ligne qu'il cherche de la forme jj/mm/aa hh:mm:ss
  while($ligne=fgets($connexion,1000)){
    print "$ligne";
    if(preg_match("/(\d\d)\/(\d\d)\/(\d\d) (\d\d):(\d\d):(\d\d)/",$ligne,$champs)){
      // on récupère les # champs
      array_shift($champs);   // enlève le 1er élément du tableau champs
      // on récupère les 6 champs dans 6 variables
      list($j,$m,$a,$h,$i,$s)=$champs;
      // affichage résultat
      print "\ndateheure=[$j,$m,$a,$h,$i,$s]\n";
    }//if
```

```
  }//while

  // le client ferme la connexion à son tour
  fclose($connexion);

  // fin
  exit;
?>
```

8.1.5 Résultats

```
HTTP/1.1 200 OK
Date: Fri, 31 Aug 2001 14:56:30 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

31/08/01 16:56:30
dateheure=[31,08,01,16,56,30]
```

8.2 Récupération par le serveur des paramètres envoyés par le client

Dans le protocole http, un client a deux méthodes pour passer des paramètres au serveur Web :

1. il demande l'URL du service sous la forme
   GET url?param1=val1&param2=val2&param3=val3… HTTP/1.0
   où les valeurs *vali* doivent être au préalable subir un encodage afin que certains caractères réservés soient remplacés par leur valeur hexadécimale.
2. il demande l'URL du service sous la forme
   POST url HTTP/1.0
   puis parmi les entêtes http envoyés au serveur place l'entête suivant :
   Content-length=N
   La suite des entêtes envoyés par le client se terminent par une ligne vide. Il peut alors envoyer ses données sous la forme
   val1&param2=val2&param3=val3…
   où les vali doivent, comme pour la méthode GET, être préalablement encodées. Le nombre de caractères envoyés au serveur doit être N où N est la valeur déclarée dans l'entête
   Content-length=N

Le script PHP qui récupère les paramètres *parami* précédents envoyés par le client obtient leurs valeurs simplement par la notation *$parami*.

8.2.1 Le client GET

```
<?
  // client : envoie prenom,nom,age au serveur par la méthode GET

  // données
  $HOTE="localhost";
  $PORT=80;
  $URL="/php/poly/paramclient/paramclient.php4";
  list($prenom,$nom,$age)=array("jean-paul","de la huche",45);

  // connexion au serveur web
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // retour si erreur
  if(! $connexion){
    print "Echec de la connexion au site ($HOTE,$PORT) : $erreur";
    exit;
  }//if
```

```
  // envoi des informations au serveur php
  // on encode les informations
  $infos="prenom=".urlencode($prenom)."&nom=" . urlencode("$nom") ."&age=" .urlencode("$age");
  print "infos envoyées au serveur (GET)=$infos\n";
  print "URL demandée=[$URL?$infos]\n\n";

  fputs($connexion, "GET $URL?$infos HTTP/1.0\n\n");

    // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
    // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
    // jusqu'à la fermeture du canal
    while($ligne=fgets($connexion,1000))
      print "$ligne";

    // le client ferme la connexion à son tour
    fclose($connexion);
?>
```

8.2.2 Le serveur

Le serveur se contente d'afficher ce qu'il reçoit.

```
<?
  // récupération par le serveur des informations envoyées par le client
  // ici prenomPp&nom=N&age=A
  // ces informations sont automatiquement disponibles dans les variables
  // $prenom, $nom, $age
  // on les renvoie au client
  print "informations reçues du client [$prenom,$nom,$age]\n";
?>
```

8.2.3 Résultats

Le serveur est installé et on lance le client qui interroge le serveur et reçoit la réponse suivante :

```
infos envoyées au serveur (GET)=prenom=jean-paul&nom=de+la+huche&age=45
URL demandée=[/php/poly/paramclient/paramclient.php4?prenom=jean-paul&nom=de+la+huche&age=45]

HTTP/1.1 200 OK
Date: Fri, 31 Aug 2001 15:45:58 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

informations reçues du client [jean-paul,de la huche,45]
```

8.2.4 Le client POST

```
<?
  // client : envoie prenom,nom,age au serveur par la méthode POST

  // données
  $HOTE="localhost";
  $PORT=80;
  $URL="/php4/documents/paramclient/paramclient.php";
  list($prenom,$nom,$age)=array("jean-paul","de la huche",45);

  // connexion au serveur web
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // retour si erreur
  if(! $connexion){
    print "Echec de la connexion au site ($HOTE,$PORT) : $erreur";
    exit;
  }//if

  // envoi des informations au serveur php
  // on encode les informations
  $infos="prenom=".urlencode($prenom)."&nom=" . urlencode("$nom") ."&age=" .urlencode("$age");
```

```
  print "client : infos envoyées au serveur (POST)=$infos\n";
  // on se connecte à l'URL $URL en lui postant (POST) des paramètres
  fputs($connexion, "POST $URL HTTP/1.0\n");
  // on indique quel type d'informations on va envoyer
  fputs($connexion,"Content-type: application/x-www-form-urlencoded\n");
  // on envoie la taille (nombre de caractères) des infos qui vont être envoyées
  fputs($connexion,"Content-length: ".strlen($infos)."\n");
  // on envoie une ligne vide
  fputs($connexion,"\n");
  // on envoie les infos
  fputs($connexion,$infos);


    // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
    // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
    // jusqu'à la fermeture du canal
    while($ligne=fgets($connexion,1000))
      print "$ligne";

    // le client ferme la connexion à son tour
    fclose($connexion);
?>
```

8.2.5 Résultats

Le serveur est installé et on lance le client qui interroge le serveur et reçoit la réponse suivante :

```
client : infos envoyées au serveur (POST)=prenom=jean-paul&nom=de+la+huche&age=45

HTTP/1.1 200 OK
Date: Fri, 31 Aug 2001 15:55:54 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

informations reçues du client [jean-paul,de la huche,45]
```

8.3 Récupération des variables d'environnement du serveur WEB

8.3.1 Le serveur

```
<?
  // environnement du serveur
  // on renvoie au client la liste des variables disponibles dans l'environnement du serveur

  while(list($clé,$valeur)=each($GLOBALS))
    print "[$clé,$valeur]\n";

?>
```

8.3.2 Le client

```
<?
  // données
  $HOTE="localhost";
  $PORT=80;
  //$urlServeur="/php4/documents/env/env.php";
  $urlServeur="/php/poly/env/env.php4";

  // ouverture d'une connexion sur le port 80 de $HOTE
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // erreur ?
  if (! $connexion){
    print "Erreur : $erreur\n";
    exit;
  }
```

```
  // on se connecte sur une URL au serveur Web
  // le client envoie la commande get pour demander l'URL /
  // syntaxe get URL HTTP/1.0
  // les entêtes (headers) du protocole HTTP doivent se terminer par une ligne vide
  fputs($connexion, "GET $urlServeur HTTP/1.0\n\n");

  // le serveur va maintenant répondre sur le canal $connexion. Il va envoyer toutes
  // ces données puis fermer le canal. Le client lit donc tout ce qui arrive de $connexion
  // jusqu'à la fermeture du canal
  while($ligne=fgets($connexion,1000)){
    print "$ligne";
  }//while

  // le client ferme la connexion à son tour
  fclose($connexion);

  // fin
  exit;

?>
```

8.3.3 Résultats

```
HTTP/1.1 200 OK
Date: Thu, 13 Sep 2001 09:34:19 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

[COMSPEC,C:\\COMMAND.COM]
[DOCUMENT_ROOT,d:/data/web]
[PATH,C:\\Program Files\\Apache Group\\Apache;C:\\Program Files\\Microsoft
Office\\Office;C:\\PROGRA~1\\MICROS~1.NET\\FRAMEW~1\\BIN\\;C:\\WINDOWS\\MICROS~1.NET\\FRAMEW~1
\\V10~1.220;C:\\JDK1.3\\BIN;C:\\ORACLE\\ORA81\\BIN;C:\\WINDOWS\\MICROSOFT.NET\\FRAMEWORK\\V1.0
.2204\\;C:\\NOVELL\\CLIENT32;C:\\WINDOWS;C:\\WINDOWS\\COMMAND;C:\\PROGRAM
FILES\\ORACLE\\JRE\\1.1.7\\BIN;C:\\PROGRAM FILES\\MICROSOFT SQL
SERVER\\80\\TOOLS\\BINN\\;Z:\\WINSHARE;Y:\\PUBLIC;X:\\BIN;W:\\TOPCAD\\BIN;]
[REDIRECT_STATUS,200]
[REDIRECT_URL,/php/poly/env/env.php4]
[REMOTE_ADDR,127.0.0.1]
[REMOTE_PORT,4606]
[SCRIPT_FILENAME,c:/php4/php.exe]
[SERVER_ADDR,127.0.0.1]
[SERVER_ADMIN,you@your.address]
[SERVER_NAME,stahe.istia.uang]
[SERVER_PORT,80]
[SERVER_SIGNATURE,<ADDRESS>Apache/1.3.12 Server at stahe.istia.uang Port 80</ADDRESS>
]
[SERVER_SOFTWARE,Apache/1.3.12 (Win32) tomcat/1.0]
[WINDIR,C:\\WINDOWS]
[GATEWAY_INTERFACE,CGI/1.1]
[SERVER_PROTOCOL,HTTP/1.0]
[REQUEST_METHOD,GET]
[QUERY_STRING,]
[REQUEST_URI,/php/poly/env/env.php4]
[SCRIPT_NAME,/php4/php.exe]
[PATH_INFO,/php/poly/env/env.php4]
[PATH_TRANSLATED,d:\\data\\web\\php\\poly\\env\\env.php4]
[PHP_SELF,/php4/php.exe/php/poly/env/env.php4]
[argv,Array]
[argc,1]
[HTTP_GET_VARS,Array]
[HTTP_COOKIE_VARS,Array]
[HTTP_SERVER_VARS,Array]
[HTTP_ENV_VARS,Array]
[GLOBALS,Array]
[valeur,Array]
[clé,valeur]
```

8.4 Gestion des sessions WEB

8.4.1 Le fichier de configuration

Pour que la gestion des sessions fonctionne correctement avec PHP, il faut vérifier que celui-ci est correctement configuré. Sous windows, son fichier de configuration est *<windows>\php.ini* où *<windows>* est le répertoire d'installation de windows. On trouve dans ce fichier une section session :

```
[Session]
session.save_handler      = files   ; handler used to store/retrieve data
session.save_path         = /temp    ; argument passed to save_handler
                                    ; in the case of files, this is the
                                    ; path where data files are stored
session.use_cookies       = 1       ; whether to use cookies
session.name              = PHPSESSID
                                    ; name of the session
                                    ; is used as cookie name
session.auto_start        = 0       ; initialize session on request startup
session.cookie_lifetime   = 0       ; lifetime in seconds of cookie
                                    ; or if 0, until browser is restarted
session.cookie_path       = /       ; the path the cookie is valid for
session.cookie_domain     =         ; the domain the cookie is valid for
session.serialize_handler = php     ; handler used to serialize data
                                    ; php is the standard serializer of PHP
session.gc_probability    = 1       ; percentual probability that the
                                    ; 'garbage collection' process is started
                                    ; on every session initialization
session.gc_maxlifetime    = 1440    ; after this number of seconds, stored
                                    ; data will be seen as 'garbage' and
                                    ; cleaned up by the gc process
session.referer_check     =         ; check HTTP Referer to invalidate
                                    ; externally stored URLs containing ids
session.entropy_length    = 0       ; how many bytes to read from the file
session.entropy_file      =         ; specified here to create the session id
; session.entropy_length    = 16
; session.entropy_file      = /dev/urandom
session.cache_limiter     = nocache ; set to {nocache,private,public} to
                                    ; determine HTTP caching aspects
session.cache_expire      = 180     ; document expires after n minutes
```

On peut garder toutes les valeurs par défaut ci-dessus mais s'assurer que le répertoire utilisé par PHP pour stocker les fichiers de suivi de session existe bien :

```
session.save_path         = /temp    ; argument passed to save_handler
                                    ; in the case of files, this is the
                                    ; path where data files are stored
```

Le répertoire ci-dessus ne précise pas de lecteur. Dans les exemples testés, il est sur le même lecteur que php. Si le répertoire ci-dessus n'existe pas, aucune erreur n'est signalée et la gestion des sessions ne fonctionne pas.

8.4.2 Le serveur 1

```
<?
  // gestion des sessions

  // on ouvre une session
  session_start();

  // on enregistre des variables afin de les suivre au fil des sessions
  if (! session_is_registered(N1)){
   session_register(N1);
   $N1=0;
  } else $N1++;
  if (! session_is_registered(N2)){
```

```
   session_register(N2);
   $N2=10;
  } else $N2++;
  if (! session_is_registered(N3)){
   session_register(N3);
   $N3=100;
  } else $N3++;

  // envoi d'informations au client
  print "N1=$N1\n";
  print "N2=$N2\n";
  print "N3=$N3\n";

  // fin de session
  session_close();
?>
```

8.4.3 Le client 1

```
<?
  // données
  $HOTE="localhost";
  $PORT=80;
  //$urlServeur="/php4/documents/sessions/sessions.php";
  $urlServeur="/php/poly/sessions/sessions.php4";

  // tests
  $cookie="";
  for($i=0;$i<5;$i++){
    list($erreur,$cookie,$N1,$N2,$N3)=connecte($HOTE,$PORT,$urlServeur,$cookie);
    print "----------------------------\n";
    print "client(erreur,cookie,N1,N2,N3)=[$erreur,$cookie,$N1,$N2,$N3]\n";
    print "----------------------------\n";
  }//if

  // fin
  exit;

  function connecte($HOTE,$PORT,$urlServeur,$cookie){
    // connecte le client à ($HOTE,$PORT,$urlServeur)
    // envoie le cookie $cookie si celui-ci est non vide
    // affiche ttes les lignes reçues en réponse

    // ouverture d'une connexion sur le port 80 de $HOTE
    $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
    // erreur ?
    if (! $connexion) return "$erreur";

    // on se connecte sur $urlserveur
    fputs($connexion, "GET $urlServeur HTTP/1.0\n");
    // on envoie le cookie s'il est non vide
    if ($cookie){
      fputs($connexion,"Cookie: $cookie\n");
    }//if
    // envoi ligne vide
    fputs($connexion,"\n");

    // on affiche la réponse du serveur web
    // et on prend soin de récupérer l'éventuel cookie et les valeurs des Ni
    $N="";
    while($ligne=fgets($connexion,1000)){
      print "$ligne";
      // cookie - seulement lors de la 1ère réponse
      if (preg_match("/^Set-Cookie: (.*?)\s*$/",$ligne,$champs)){
        $cookie=$champs[1];
      }//if
      // valeur de N1
      if (preg_match("/^N1=(.*?)\s*$/",$ligne,$champs))
        $N1=$champs[1];
      // valeur de N2
      if (preg_match("/^N2=(.*?)\s*$/",$ligne,$champs))
        $N2=$champs[1];
      // valeur de N1
      if (preg_match("/^N3=(.*?)\s*$/",$ligne,$champs))
        $N3=$champs[1];
```

```
    }//while

    // on ferme la connexion
    fclose($connexion);
    // retour
    return array("",$cookie,$N1,$N2,$N3);
  }//connecte
?>
```

8.4.4 Résultats

```
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:48:17 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Set-Cookie: PHPSESSID=85fab1c94611505929e2875af9082bce
Connection: close
Content-Type: text/html

N1=0
N2=10
N3=100
----------------------------
client(erreur,cookie,N1,N2,N3)=[,PHPSESSID=85fab1c94611505929e2875af9082bce,0,10,100]
----------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:48:18 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

N1=1
N2=11
N3=101
----------------------------
client(erreur,cookie,N1,N2,N3)=[,PHPSESSID=85fab1c94611505929e2875af9082bce,1,11,101]
----------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:48:18 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

N1=2
N2=12
N3=102
----------------------------
client(erreur,cookie,N1,N2,N3)=[,PHPSESSID=85fab1c94611505929e2875af9082bce,2,12,102]
----------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:48:18 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

N1=3
N2=13
N3=103
----------------------------
client(erreur,cookie,N1,N2,N3)=[,PHPSESSID=85fab1c94611505929e2875af9082bce,3,13,103]
```

```
----------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:48:21 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

N1=4
N2=14
N3=104
----------------------------
client(erreur,cookie,N1,N2,N3)=[,PHPSESSID=85fab1c94611505929e2875af9082bce,4,14,104]
----------------------------
```

On voit clairement que le serveur Web conserve les valeurs de (N1,N2,N3) au fil des demandes du client. C'est ce qu'on appelle le suivi de session. Les deux exemples qui suivent montrent qu'on peut ainsi sauvegarder les valeurs d'un tableau ou d'un objet.

8.4.5 Le serveur 2

```
<?
  // gestion des sessions

  // on ouvre une session
  session_start();

  // on enregistre un tableau et un dictionnaire
  if (! session_is_registered(tableau)) session_register(tableau);
  if (! session_is_registered(dico)) session_register(dico);

  // on initialise ou modifie le tableau
  if (isset($tableau))
    for($i=0;$i<count($tableau);$i++)
      $tableau[$i]++;
  else
    for($i=0;$i<10;$i++)
      $tableau[$i]=$i*10;

  // on initialise ou modifie le dictionnaire
  if (isset($dico)){
    $clés=array_keys($dico);
    for($i=0;$i<count($clés);$i++)
      $dico[$clés[$i]]++;
  }
  else
    $dico=array("zéro"=>0,"dix"=>10,"vingt"=>20);

  // envoi d'informations au client
  print "tableau=".join(",",$tableau)."\n";
  print "dico=";
  while (list($clé,$valeur)=each($dico))
    print "($clé,$valeur) ";
  print "\n";
?>
```

8.4.6 Le client 2

```
<?
  // données
  $HOTE="localhost";
  $PORT=80;
  $urlServeur="/php/poly/sessions/sessions2.php4";

  // tests
  $cookie="";
  for($i=0;$i<5;$i++)
    connecte($HOTE,$PORT,$urlServeur,&$cookie);
```

```
  // fin
  exit;

  function connecte($HOTE,$PORT,$urlServeur,$cookie){
    // connecte le client à ($HOTE,$PORT,$urlServeur)
    // envoie le cookie $cookie si celui-ci est non vide
    // affiche ttes les lignes reçues en réponse

    // séparateurs pour l'affichage
    print "-------------------------------------------------------------\n";

    // ouverture d'une connexion sur le port 80 de $HOTE
    $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
    // erreur ?
    if (! $connexion) return "$erreur";

    // on se connecte sur $urlserveur
    fputs($connexion, "GET $urlServeur HTTP/1.0\n");
    // on envoie le cookie s'il est non vide
    if ($cookie){
      fputs($connexion,"Cookie: $cookie\n");
    }//if
    // envoi ligne vide
    fputs($connexion,"\n");

    // on affiche la réponse du serveur web
    // et on prend soin de récupérer l'éventuel cookie et les valeurs des Ni
    $N="";
    while($ligne=fgets($connexion,1000)){
      print "$ligne";
      // cookie - seulement lors de la 1ère réponse
      if (preg_match("/^Set-Cookie: (.*?)\s*$/",$ligne,$champs)){
        $cookie=$champs[1];
      }//if
    }//while

    // on ferme la connexion
    fclose($connexion);
    // retour
    return "";
  }//connecte
?>
```

8.4.7 Résultats

```
-------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:57:19 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Set-Cookie: PHPSESSID=06bd99ab0681ebd718903649abba85b6
Connection: close
Content-Type: text/html

tableau=0,10,20,30,40,50,60,70,80,90
dico=(zéro,0) (dix,10) (vingt,20)
-------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:57:19 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

tableau=1,11,21,31,41,51,61,71,81,91
dico=(zéro,1) (dix,11) (vingt,21)
-------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:57:20 GMT
```

```
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

tableau=2,12,22,32,42,52,62,72,82,92
dico=(zéro,2) (dix,12) (vingt,22)
------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:57:20 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

tableau=3,13,23,33,43,53,63,73,83,93
dico=(zéro,3) (dix,13) (vingt,23)
------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 06:57:21 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

tableau=4,14,24,34,44,54,64,74,84,94
dico=(zéro,4) (dix,14) (vingt,24)
```

8.4.8 Le serveur 3

Ici, on mémorise la valeur d'un objet.

```
<?
  // gestion des sessions

  // on ouvre une session
  session_start();

  // on enregistre un objet
  if (! session_is_registered(personne)) session_register(personne);

  // on initialise ou modifie l'objet
  if (! isset($personne)){
    $personne->nom="dupont";
    $personne->age=30;
  }
  else
    $personne->age++;

  // envoi d'informations au client
  print "personne=($personne->nom,$personne->age)\n";;
?>
```

8.4.9 Le client 3

```
<?
  // données
  $HOTE="localhost";
  $PORT=80;
  $urlServeur="/php/poly/sessions/sessions3.php4";

  // tests
  $cookie="";
  for($i=0;$i<5;$i++)
```

```
    connecte($HOTE,$PORT,$urlServeur,&$cookie);

  // fin
  exit;

  function connecte($HOTE,$PORT,$urlServeur,$cookie){
    // connecte le client à ($HOTE,$PORT,$urlServeur)
    // envoie le cookie $cookie si celui-ci est non vide
    // affiche ttes les lignes reçues en réponse

    // séparateurs pour l'affichage
    print "-----------------------------------------------------------\n";

    // ouverture d'une connexion sur le port 80 de $HOTE
    $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
    // erreur ?
    if (! $connexion) return "$erreur";

    // on se connecte sur $urlserveur
    fputs($connexion, "GET $urlServeur HTTP/1.0\n");
    // on envoie le cookie s'il est non vide
    if ($cookie){
      fputs($connexion,"Cookie: $cookie\n");
    }//if
    // envoi ligne vide
    fputs($connexion,"\n");

    // on affiche la réponse du serveur web
    // et on prend soin de récupérer l'éventuel cookie et les valeurs des Ni
    $N="";
    while($ligne=fgets($connexion,1000)){
      print "$ligne";
      // cookie - seulement lors de la 1ère réponse
      if (preg_match("/^Set-Cookie: (.*?)\s*$/",$ligne,$champs)){
        $cookie=$champs[1];
      }//if
    }//while

    // on ferme la connexion
    fclose($connexion);
    // retour
    return "";
  }//connecte
?>
```

8.4.10 Résultats

```
-----------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 07:02:02 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Set-Cookie: PHPSESSID=dff33675d926696fb8be7e5496cbf31d
Connection: close
Content-Type: text/html

personne=(dupont,30)
-----------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 07:02:03 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

personne=(dupont,31)
-----------------------------------------------------------
HTTP/1.1 200 OK
```

```
Date: Fri, 14 Sep 2001 07:02:03 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

personne=(dupont,32)
------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 07:02:04 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

personne=(dupont,33)
------------------------------------------------------------
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 07:02:04 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
Cache-Control: no-cache, post-check=0, pre-check=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

personne=(dupont,34)
```

9 Traiter l'exercice IMPOTS avec un service WEB

9.1 Le serveur

```
<?
  // le service web des impôts

  // définition des constantes
  $TABLE="impots";
  $BASE="impots";
  $USER="admimpots";
  $PWD="mdpimpots";

  // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL $TABLE
  // appartenant à la base $BASE. La table a la structure suivante
  // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
  // les paramètres des personnes imposables (statut marital, nombre d'enfants, salaire
annuel)
  // sont envoyés par le client sous la forme params=statut marital, nombre d'enfants, salaire
annuel
  // les résultats (statut marital, nombre d'enfants, salaire annuel, impôt à payer) sont
renvoyés au client
  // sous la forme 200 impot
  // ou sous la forme 500 erreur, si les paramètres sont invalides

  // on crée un objet impôts
  $I=new impôts($HOTE,$USER,$PWD,$BASE,$TABLE);
  // y-a-t-il eu une erreur ?
  if($I->erreur){
    print "$I->erreur\n";
    exit;
  }//if

  // on crée un objet utilitaires
  $u=new utilitaires();

  // on récupère la ligne envoyée par le client
  print "paramètres reçus --> $params\n";
  $params=strtolower(trim($params));
  $items=explode(",",$params);
  // il ne doit y avoir que 3 paramètres
  if (count($items)!=3){
    print "<erreur>[$params] : nombre de paramètres invalides</erreur>\n";
    exit;
  }//if
  // le premier paramètre (statut marital) doit être oui/non
  $marié=trim($items[0]);
  if ($marié!="oui" and $marié != "non"){
    print "<erreur>[$params] : 1er paramètre invalide</erreur>\n";
    exit;
  }//if
  // le second paramètre (nbre d'enfants) doit être un nombre entier
  if (! preg_match("/^\s*(\d+)\s*$/",$items[1],$champs)){
    print "<erreur>[$params] : 2ième paramètre invalide</erreur>\n";
    exit;
  }//if
  $enfants=$champs[1];
  // le troisième paramètre (salaire) doit être un nombre entier
  if (! preg_match("/^\s*(\d+)\s*$/",$items[2],$champs)){
    print "<erreur>[$params] : 3ième paramètre invalide</erreur>\n";
    exit;
  }//if
  $salaire=$champs[1];

  // on calcule l'impôt
  $impôt=$I->calculer($marié,$enfants,$salaire);
  // on renvoie le résultat
  print "<impot>$impôt</impot>\n";

  // fin
  exit;

  // ----------------------------------------------------------------------------
```

```
  // définition d'une classe dataImpôts
  class impôts{
    // attributs : les 3 tableaux de données
    var $limites;
    var $coeffR;
    var $coeffN;
    var $erreur;
    var $nbLimites; // nbre d'éléments dans tableau $limites

    // constructeur
    function impôts($HOTE,$USER,$PWD,$BASE,$TABLE){
      // initialise les attributs $limites, $coeffR, $coeffN
      // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL
$TABLE
      // appartenant à la base $BASE. La table a la structure suivante
      // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
      // la connexion à la base mysql de la machine $HOTE se fait sous l'identité ($USER,$PWD)
      // rend une chaîne $erreur indiquant la nature d'une éventuelle erreur
      // vide si pas d'erreur

      // connexion à la base mysql
        list($erreur,$connexion)=connecte($HOTE,$USER,$PWD);
        if ($erreur){
          $this->erreur="Erreur lors de la connexion à MySql sous l'identité
($HOTE,$USER,$PWD) : $erreur\n";
          return;
        }
      // lecture en bloc de la table $TABLE
        $requête="select limites,coeffR,coeffN from $TABLE";
        // exécute la requête $requête sur la base $base de la connexion $connexion
        if (! ($résultat=@mysql_db_query($BASE,$requête,$connexion))){
           $this->erreur="$requête : ".mysql_error();
           return;
        }//if
        // exploitation du résultat de la requête
        while($colonnes=mysql_fetch_row($résultat)){
          // les nombres mysql sont de la forme eee.dd
          // alors que php4/windows veut eee,dd à cause d'un bogue
          for($i=0;$i<count($colonnes);$i++)
            $colonnes[$i]=str_replace(".",",",$colonnes[$i]);
          $this->limites[]=$colonnes[0];
          $this->coeffR[]=$colonnes[1];
          $this->coeffN[]=$colonnes[2];
        }//while
      // déconnexion
      déconnecte($connexion);
      // pas d'erreur
      $this->erreur="";
      // nombre d'élements de limites
      $this->nbLimites=count($this->limites);
    }//constructeur impôts

    // ------------------------------------------------------------------------
    function calculer($marié,$enfants,$salaire){
      // $marié : oui, non
      // $enfants : nombre d'enfants
      // $salaire : salaire annuel

      // l'objet est-il dans un état correct ?
      if ($this->erreur) return -1;

      // nombre de parts
      $marié=strToLower($marié);
      if($marié=="oui") $nbParts=$enfants/2+2;
        else $nbParts=$enfants/2+1;
      // une 1/2 part de plus si au moins 3 enfants
      if($enfants>=3) $nbParts+="0,5";
      // revenu imposable
      $revenuImposable="0,72"*$salaire;
      // quotient familial
      $quotient=$revenuImposable/$nbParts;
      // est mis à la fin du tableau limites pour arrêter la boucle qui suit
      $N=$this->nbLimites;
      $this->limites[$N-1]=$quotient;
      // calcul de l'impôt
      $i=0;
      while($i<$this->nbLimites and $quotient>$this->limites[$i]) $i++;
```

```
      // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
      // ne peut déborder du tableau $limites
      // maintenant on peut calculer l'impôt
      return floor($revenuImposable*$this->coeffR[$i]-$nbParts*$this->coeffN[$i]);
    }//calculImpots
  }//classe impôts

  // une classe de fonctions utilitaires
  class utilitaires{
    function cutNewLinechar($ligne){
    // on supprime la marque de fin de ligne de $ligne si elle existe
      $L=strlen($ligne);  // longueur ligne
      while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
        $ligne=substr($ligne,0,$L-1);
        $L--;
      }//while
      // fin
      return($ligne);
    }//cutNewLineChar
  }//classe utilitaires

 // ------------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête

?>
```

9.2 Le client

```
<?
  // client impôts
  $instructions=array("oui,2,200000","non,2,200000","oui,3,200000","non,3,200000","x,y,z,t","x
,2,200000", "oui,x,200000",
    "oui,2,x");

  // données serveur
  $HOTE="localhost";
  $PORT=80;
  $urlServeur="/php/poly/impots/impots.php4";

  // tests
  $cookie="";
  for($i=0;$i<count($instructions);$i++)
    connecte($HOTE,$PORT,$urlServeur,$instructions[$i],&$cookie);

  // fin
  exit;

  function connecte($HOTE,$PORT,$urlServeur,$instruction,$cookie){
    // connecte le client à ($HOTE,$PORT,$urlServeur)
    // envoie le cookie $cookie si celui-ci est non vide
    // envoie $instruction au serveur
    // affiche ttes les lignes reçues en réponse
```

```
    // séparateurs pour l'affichage
    print "-------------------------------------------------------------\n";

    // ouverture d'une connexion sur le port 80 de $HOTE
    $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
    // erreur ?
    if (! $connexion) return "$erreur";

    // on se connecte sur $urlserveur
    fputs($connexion, "POST $urlServeur HTTP/1.0\n");
    // on envoie le cookie s'il est non vide
    if ($cookie){
      fputs($connexion,"Cookie: $cookie\n");
    }//if
    // maintenant on envoie l'instruction client après l'avoir encodée
    $infos="params=".urlencode($instruction);
    print "client : infos envoyées au serveur (POST)=[$infos]\n";
    // on indique quel type d'informations on va envoyer
    fputs($connexion,"Content-type: application/x-www-form-urlencoded\n");
    // on envoie la taille (nombre de caractères) des infos qui vont être envoyées
    fputs($connexion,"Content-length: ".strlen($infos)."\n");
    // on envoie une ligne vide
    fputs($connexion,"\n");
    // on envoie les infos
    // suivi
    fputs($connexion,$infos);

    // on affiche la réponse du serveur web
    // et on prend soin de récupérer l'éventuel cookie
    while($ligne=fgets($connexion,1000)){
      print "$ligne";
      // cookie - seulement lors de la 1ère réponse
      if (preg_match("/^Set-Cookie: (.*?)\s*$/",$ligne,$champs)){
        $cookie=$champs[1];
      }//if
      // dès qu'on a un nombre a 3 chiffres c'est la réponse du serveur
      if (preg_match("/^(\d\d\d) (.*?)\s*$/",$ligne,$champs)){
        array_shift($champs);
        list($code,$résultat)=$champs;
      }//if
    }//while

    // on ferme la connexion
    fclose($connexion);
    // affichage résultat
    if($code=="200")
      print "impôt=$résultat\n";
    else print "erreur=$résultat\n";

    // retour
    return "";
  }//connecte
?>
```

9.3 Résultats

```
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:24 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> oui,2,200000
200 22506
impôt=22506
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=non%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:25 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
```

```
Connection: close
Content-Type: text/html

paramètres reçus --> non,2,200000
200 33388
impôt=33388
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C3%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:25 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> oui,3,200000
200 16400
impôt=16400
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=non%2C3%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:26 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> non,3,200000
200 22506
impôt=22506
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=x%2Cy%2Cz%2Ct]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:26 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> x,y,z,t
500 [x,y,z,t] : nombre de paramètres invalides
erreur=[x,y,z,t] : nombre de paramètres invalides
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=x%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:26 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> x,2,200000
501 [x,2,200000] : 1er paramètre invalide
erreur=[x,2,200000] : 1er paramètre invalide
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2Cx%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:27 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> oui,x,200000
500 [oui,x,200000] : 2ième paramètre invalide
erreur=[oui,x,200000] : 2ième paramètre invalide
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C2%2Cx]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:10:27 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

paramètres reçus --> oui,2,x
500 [oui,2,x] : 3ième paramètre invalide
```

```
erreur=[oui,2,x] : 3ième paramètre invalide
```

10 Traitement de documents XML

Nous considérons le document XML suivant :

```
<tribu>
  <enseignant>
    <personne sexe="M">
      <nom>dupont</nom>
      <prenom>jean</prenom>
      <age>28</age>
      ceci est un commentaire
    </personne>
    <section>27</section>
  </enseignant>
  <etudiant>
    <personne sexe="F">
      <nom>martin</nom>
      <prenom>charline</prenom>
      <age>22</age>
    </personne>
    <formation>dess IAIE</formation>
  </etudiant>
</tribu>
```

Nous analysons ce document avec le programme suivant :

```
<?
  // données
  $file="data.xml";       // le fichier xml
  $depth=0;               // niveau d'indentation=profondeur dans l'arborescence
  $balisesDonnées=array("nom"=>1,"prenom"=>1,"age"=>1,"section"=>1,"formation"=>1);
  $baliseDeDonnées=0;     // à vrai, indique qu'on a affaire à une balise de données

  // fonction appelée lors de la rencontre d'une balise de début
  function startElement($parser,$name,$attributs){
    global $depth;
    // une suite d'espaces (indentation)
    for($i=0;$i<$depth;$i++){
      print " ";
    }//for
    // attributs
    $précisions="";
    while(list($attrib,$valeur)=each($attributs)){
      $précisions.="($attrib,$valeur) ";
    }
    // on affiche le nom de la balise et les éventuels attributs
    if($précisions)
      print "$name,$précisions\n";
    else print "$name\n";
    // un niveau d'arborescence en plus
    $depth++;
    // est-ce une balise de données ?
    global $balisesDonnées,$baliseDeDonnées;
    if ($balisesDonnées[strtolower($name)]){
      $baliseDeDonnées=1;
    }//if
  }//startElement

  // la fonction appelée lors de la rencontre d'une balise de fin
  function endElement($parser,$name){
    // fin de balise
    // niveau d'indentation
    global $depth;
    $depth--;
    // une suite d'espaces (indentation)
    for($i=0;$i<$depth;$i++){
      print " ";
    }//for
    // le nom de la balise
    print "/$name\n";
  }//endElement
```

```
  // la fonction d'affichage des données
  function afficheData($parser,$data){
    // données
    global $baliseDeDonnées;

    // la balise courante est-elle une balise de données ?
    if (! $baliseDeDonnées) return;
    // niveau d'indentation
    global $depth;
    // une suite d'espaces (indentation)
    for($i=0;$i<$depth;$i++){
      print " ";
    }//for
    // les données sont affichées
    preg_match("/^\s*(.*?)\s*$/",$data,$champs);
    print "[".$champs[1]."]\n";
    // fin de la balise de données
    $baliseDeDonnées=0;
  }//afficheData

  // le programme

  // on crée un objet d'analyse de texte xml
  $xml_parser=xml_parser_create();
  // on indique quelles fonctions exécuter en début et fin de balise
  xml_set_element_handler($xml_parser,"startElement","endElement");
  // on indique quelle fonction exécuter lors de la rencontre de données
  xml_set_character_data_handler($xml_parser,"afficheData");
  // ouverture du fichier xml en lecture
  if (! ($fp=@fopen($file,"r"))){
    print "impossible d'ouvrir le fichier xml $file";
    exit;
  }//if
  // exploitation du fichier xml
  // par blocs de 4096 octets
  while($data=fread($fp,4096)){
    // analyse des données lues
    if (! xml_parse($xml_parser,$data,feof($fp))){
      // il s'est produit une erreur
      printf ("erreur XML : %s à la ligne %d\n",
        xml_error_string(xml_get_error_code($xml_parser)),
        xml_get_current_line_number($xml_parser));
      // fin
      exit;
    }//if
  }//while
  // le fichier a été exploré
  // on libère les ressources occupées par l'analyseur xml
  xml_parser_free($xml_parser);
  // fin
  exit;
?>
```

Nous obtenons le résultat suivant :

```
TRIBU
 ENSEIGNANT
  PERSONNE,(SEXE,M)
   NOM
    [dupont]
   /NOM
   PRENOM
    [jean]
   /PRENOM
   AGE
    [28]
   /AGE
  /PERSONNE
  SECTION
   [27]
  /SECTION
 /ENSEIGNANT
 ETUDIANT
  PERSONNE,(SEXE,F)
   NOM
```

```
    [martin]
   /NOM
   PRENOM
    [charline]
   /PRENOM
   AGE
    [22]
   /AGE
  /PERSONNE
  FORMATION
   [dess IAIE]
  /FORMATION
 /ETUDIANT
/TRIBU
```

11 Exercice IMPOTS avec XML

Dans cet exercice déjà étudié de nombreuses fois, le serveur renvoie les résultats au client sous la forme d'un flux XML.

11.1 Le serveur

```
<?
  // le service web des impôts

  // définition des constantes
  $TABLE="impots";
  $BASE="impots";
  $USER="admimpots";
  $PWD="mdpimpots";

  // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL $TABLE
  // appartenant à la base $BASE. La table a la structure suivante
  // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
  // les paramètres des personnes imposables (statut marital, nombre d'enfants, salaire annuel)
  // sont envoyés par le client sous la forme params=statut marital, nombre d'enfants, salaire annuel
  // les résultats (statut marital, nombre d'enfants, salaire annuel, impôt à payer) sont renvoyés au
client
  // sous la forme 200 impot
  // ou sous la forme 500 erreur, si les paramètres sont invalides

  // on crée un objet impôts
  $I=new impôts($HOTE,$USER,$PWD,$BASE,$TABLE);
  // y-a-t-il eu une erreur ?
  if($I->erreur){
    print "$I->erreur\n";
    exit;
  }//if

  // on crée un objet utilitaires
  $u=new utilitaires();

  // on récupère la ligne envoyée par le client et on la renvoie
  print "<reponse><parametres>$params</parametres>\n";
  $params=strtolower(trim($params));
  $items=explode(",",$params);
  // il ne doit y avoir que 3 paramètres
  if (count($items)!=3){
    print "<erreur>[$params] : nombre de paramètres invalides</erreur></reponse>\n";
    exit;
  }//if
  // le premier paramètre (statut marital) doit être oui/non
  $marié=trim($items[0]);
  if ($marié!="oui" and $marié != "non"){
    print "<erreur>[$params] : 1er paramètre invalide</erreur></reponse>\n";
    exit;
  }//if
  // le second paramètre (nbre d'enfants) doit être un nombre entier
  if (! preg_match("/^\s*(\d+)\s*$/",$items[1],$champs)){
    print "<erreur>[$params] : 2ième paramètre invalide</erreur></reponse>\n";
    exit;
  }//if
  $enfants=$champs[1];
  // le troisième paramètre (salaire) doit être un nombre entier
  if (! preg_match("/^\s*(\d+)\s*$/",$items[2],$champs)){
    print "<erreur>[$params] : 3ième paramètre invalide</erreur></reponse>\n";
    exit;
  }//if
  $salaire=$champs[1];

  // on calcule l'impôt
  $impôt=$I->calculer($marié,$enfants,$salaire);
  // on renvoie le résultat
  print "<impot>$impôt</impot></reponse>\n";
```

```
  // fin
  exit;

  // ------------------------------------------------------------------------------
  // définition d'une classe dataImpôts
  class impôts{
    // attributs : les 3 tableaux de données
    var $limites;
    var $coeffR;
    var $coeffN;
    var $erreur;
    var $nbLimites; // nbre d'éléments dans tableau $limites

    // constructeur
    function impôts($HOTE,$USER,$PWD,$BASE,$TABLE){
      // initialise les attributs $limites, $coeffR, $coeffN
      // les données nécessaires au calcul de l'impôt ont été placées dans la table mysqL $TABLE
      // appartenant à la base $BASE. La table a la structure suivante
      // limites decimal(10,2), coeffR decimal(6,2), coeffN decimal(10,2)
      // la connexion à la base mysql de la machine $HOTE se fait sous l'identité ($USER,$PWD)
      // rend une chaîne $erreur indiquant la nature d'une éventuelle erreur
      // vide si pas d'erreur

      // connexion à la base mysql
        list($erreur,$connexion)=connecte($HOTE,$USER,$PWD);
        if ($erreur){
          $this->erreur="Erreur lors de la connexion à MySql sous l'identité ($HOTE,$USER,$PWD) :
$erreur\n";
          return;
        }
      // lecture en bloc de la table $TABLE
        $requête="select limites,coeffR,coeffN from $TABLE";
        // exécute la requête $requête sur la base $base de la connexion $connexion
        if (! ($résultat=@mysql_db_query($BASE,$requête,$connexion))){
           $this->erreur="$requête : ".mysql_error();
           return;
        }//if
        // exploitation du résultat de la requête
        while($colonnes=mysql_fetch_row($résultat)){
          // les nombres mysql sont de la forme eee.dd
          // alors que php4/windows veut eee,dd à cause d'un bogue
          for($i=0;$i<count($colonnes);$i++)
            $colonnes[$i]=str_replace(".",",",$colonnes[$i]);
          $this->limites[]=$colonnes[0];
          $this->coeffR[]=$colonnes[1];
          $this->coeffN[]=$colonnes[2];
        }//while
      // déconnexion
      déconnecte($connexion);
      // pas d'erreur
      $this->erreur="";
      // nombre d'élements de limites
      $this->nbLimites=count($this->limites);
    }//constructeur impôts

    // ------------------------------------------------------------------------------
    function calculer($marié,$enfants,$salaire){
      // $marié : oui, non
      // $enfants : nombre d'enfants
      // $salaire : salaire annuel

      // l'objet est-il dans un état correct ?
      if ($this->erreur) return -1;

      // nombre de parts
      $marié=strToLower($marié);
      if($marié=="oui") $nbParts=$enfants/2+2;
        else $nbParts=$enfants/2+1;
      // une 1/2 part de plus si au moins 3 enfants
      if($enfants>=3) $nbParts+="0,5";
      // revenu imposable
      $revenuImposable="0,72"*$salaire;
      // quotient familial
      $quotient=$revenuImposable/$nbParts;
      // est mis à la fin du tableau limites pour arrêter la boucle qui suit
```

```
      $N=$this->nbLimites;
      $this->limites[$N-1]=$quotient;
      // calcul de l'impôt
      $i=0;
      while($i<$this->nbLimites and $quotient>$this->limites[$i]) $i++;
      // du fait qu'on a placé $quotient à la fin du tableau $limites, la boucle précédente
      // ne peut déborder du tableau $limites
      // maintenant on peut calculer l'impôt
      return floor($revenuImposable*$this->coeffR[$i]-$nbParts*$this->coeffN[$i]);
    }//calculImpots
  }//classe impôts

  // une classe de fonctions utilitaires
  class utilitaires{
    function cutNewLinechar($ligne){
    // on supprime la marque de fin de ligne de $ligne si elle existe
      $L=strlen($ligne);  // longueur ligne
      while(substr($ligne,$L-1,1)=="\n" or substr($ligne,$L-1,1)=="\r"){
        $ligne=substr($ligne,0,$L-1);
        $L--;
      }//while
      // fin
      return($ligne);
    }//cutNewLineChar
  }//classe utilitaires

 // ------------------------------------------------------------------------------
  function connecte($mysql,$login,$pwd){
    // connecte ($login,$pwd) à la base mysql $mysql
    // rend l'id de la connexion ainsi qu'un code d'erreur
    $connexion=@mysql_connect($mysql,$login,$pwd);
    if ($connexion) return array("",$connexion);
      else return array($php_errormsg);
  }//connecte

  // ------------------------------------------------------------------------------
  function déconnecte($connexion){
    // ferme la connexion mysql identifiée par $connexion
    // rend un code d'erreur
    if (@mysql_close($connexion)) return "";
      else return mysql_error;
  }//ferme

  // ------------------------------------------------------------------------------
  function exécuteRequête($connexion,$base,$sql){
    // exécute la requête $requête sur la base $base de la connexion $connexion
    if ($res=@mysql_db_query($base,$sql,$connexion))
      return array("",$res);
    else return array(mysql_error());
  }//exécuteRequête

?>
```

11.2 Le client

```
<?
  // client impôts
  $instructions=array("oui,2,200000","non,2,200000","oui,3,200000","non,3,200000","x,y,z,t","x,2,200000",
"oui,x,200000",
    "oui,2,x");

  // données serveur
  $HOTE="localhost";
  $PORT=80;
  //$urlServeur="/php4/documents/impots/impotsxml.php";
  $urlServeur="/php/poly/impotsxml/impotsxml.php4";

  // tests
  $cookie="";
  for($i=0;$i<count($instructions);$i++)
    connecte($HOTE,$PORT,$urlServeur,$instructions[$i],&$cookie);

  // fin
  exit;
```

```php
function connecte($HOTE,$PORT,$urlServeur,$instruction,$cookie){
  // connecte le client à ($HOTE,$PORT,$urlServeur)
  // envoie le cookie $cookie si celui-ci est non vide
  // envoie $instruction au serveur
  // affiche ttes les lignes reçues en réponse

  // séparateurs pour l'affichage
  print "-------------------------------------------------------------\n";

  // ouverture d'une connexion sur le port 80 de $HOTE
  $connexion=fsockopen($HOTE,$PORT,&$errno,&$erreur);
  // erreur ?
  if (! $connexion) return "$erreur";

  // on se connecte sur $urlserveur
  fputs($connexion, "POST $urlServeur HTTP/1.0\n");
  // on envoie le cookie s'il est non vide
  if ($cookie){
    fputs($connexion,"Cookie: $cookie\n");
  }//if
  // maintenant on envoie l'instruction client après l'avoir encodée
  $infos="params=".urlencode($instruction);
  print "client : infos envoyées au serveur (POST)=[$infos]\n";
  // on indique quel type d'informations on va envoyer
  fputs($connexion,"Content-type: application/x-www-form-urlencoded\n");
  // on envoie la taille (nombre de caractères) des infos qui vont être envoyées
  fputs($connexion,"Content-length: ".strlen($infos)."\n");
  // on envoie une ligne vide
  fputs($connexion,"\n");
  // on envoie les infos
  // suivi
  fputs($connexion,$infos);

  // on affiche l'entête de la réponse du serveur web
  while(($ligne=fgets($connexion,1000)) && (! preg_match("/^\s*$/",$ligne))){
    // suivi
    print "$ligne";
    // cookie - seulement lors de la 1ère réponse
    if (preg_match("/^Set-Cookie: (.*?)\s*$/",$ligne,$champs)){
      $cookie=$champs[1];
    }//if
  }//while
  // suite de la réponse du serveur web
  print "$ligne";
  $réponse="";
  while($ligne=fgets($connexion,1000)){
    // suivi
    print "$ligne";
    // on concatène la ligne à la réponse en cours de constitution
    $réponse.=$ligne;
  }//while
  // on ferme la connexion
  fclose($connexion);
  // on analyse la réponse
  list($erreur,$impôt)=getrésultat($réponse);
  // affichage résultat
  if(! $erreur)
    print "impôt=$impôt\n";
  else print "erreur=$erreur\n";
  // retour
  return "";
}//connecte

// -------------------------------------------------------------
function getrésultat($réponse){
  // $réponse est de la forme
  // <erreur>texte</erreur>
  // ou <impot>texte</impot>

  // données
  global $elements,$elementcourant;
  $elements=array();
  $elementcourant="";
```

```
    // on crée un objet d'analyse de texte xml
    $xml_parser=xml_parser_create();
    // on indique quelles fonctions exécuter en début et fin de balise
    xml_set_element_handler($xml_parser,"startElement","endElement");
    // on indique quelle fonction exécuter lors de la rencontre de données
    xml_set_character_data_handler($xml_parser,"getData");
    // exploitation de la réponse xml
    if (! xml_parse($xml_parser,$réponse,1)){
      // il s'est produit une erreur
      printf ("erreur XML : %s à la ligne %d colonne %d\n",
        xml_error_string(xml_get_error_code($xml_parser)),
        xml_get_current_line_number($xml_parser),
        xml_get_current_column_number($xml_parser));
    }

    // on libère les ressources occupées par l'analyseur xml
    xml_parser_free($xml_parser);
    // fin - on rend les résultats
    return array($elements["ERREUR"],$elements["IMPOT"]);
  }//getrésultat

  // fonction appelée lors de la rencontre d'une balise de début
  function startElement($parser,$name,$attributs){
    // on note l'élément courant
    global $elementcourant;
    $elementcourant=$name;
  }//startElement

  // fonction appelée lors de la rencontre d'une balise de fin
  function endElement($parser,$name){
    // ne fait rien
  }//endElement

  // la fonction d'obtention des données
  function getData($parser,$data){
    // si $data est non vide, on l'associe à l'élément courant
    global $elements,$elementcourant;
    if (preg_match("/^\s*(.+?)\s*$/",$data,$champs)){
      $elements[$elementcourant]=$champs[1];
    }//if
  }//afficheData

?>
```

11.3 Les résultats

```
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:07 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>oui,2,200000</parametres>
<impot>22506</impot></reponse>
impôt=22506
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=non%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:08 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>non,2,200000</parametres>
<impot>33388</impot></reponse>
impôt=33388
-------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C3%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:08 GMT
```

```
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>oui,3,200000</parametres>
<impot>16400</impot></reponse>
impôt=16400
------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=non%2C3%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:09 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>non,3,200000</parametres>
<impot>22506</impot></reponse>
impôt=22506
------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=x%2Cy%2Cz%2Ct]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:09 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>x,y,z,t</parametres>
<erreur>[x,y,z,t] : nombre de paramètres invalides</erreur></reponse>
erreur=[x,y,z,t] : nombre de paramètres invalides
------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=x%2C2%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:10 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>x,2,200000</parametres>
<erreur>[x,2,200000] : 1er paramètre invalide</erreur></reponse>
erreur=[x,2,200000] : 1er paramètre invalide
------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2Cx%2C200000]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:11 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>oui,x,200000</parametres>
<erreur>[oui,x,200000] : 2ième paramètre invalide</erreur></reponse>
erreur=[oui,x,200000] : 2ième paramètre invalide
------------------------------------------------------------
client : infos envoyées au serveur (POST)=[params=oui%2C2%2Cx]
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2001 08:25:11 GMT
Server: Apache/1.3.12 (Win32) tomcat/1.0
X-Powered-By: PHP/4.0RC2
Connection: close
Content-Type: text/html

<reponse><parametres>oui,2,x</parametres>
<erreur>[oui,2,x] : 3ième paramètre invalide</erreur></reponse>
erreur=[oui,2,x] : 3ième paramètre invalide
```