Master Intégration de compétences - "Introduction Système et Réseau" - FMIN112
Pierre Pompidor - vendredi 15 janvier 2010 (et non 2009) - Correction

Compréhension d'un script système écrit en Perl : 11 points

Expression régulière 1 : `$exp =~ /^\*(.*)/`
elle permet de capturer une chaîne de caractères suivant une étoile.

Expression régulière 2 : `$exp =~ /(.*[^\.])\*(.*)/`
elle permet de capturer une chaîne de caractères suivant une étoile non précédée par un point.

En fait avant d'appeler la fonction récursive parcours, le second paramètre qui a été recopié dans la variable $exp est modifié pour que les * soient transformées par des .* .

La fonction parcours ouvre le répertoire dont le nom lui est passé en premier paramètre.

Expression régulière 3 : `$fichier !~ /^\./`
vérifie que `$fichier` ne commence pas par un point : cette vérification va permettre de ne pas prendre en compte les fichiers cachés et les deux répertoires . et ..

La fonction se rappelle récursivement sur tous les sous-répertoires du répertoire initial.

Expression régulière 4 : `$fichier =~ /^$f$/`
vérifie que le nom du fichier correspond au contenu de la variable `$f` ; comme nous avons vu que cette variable contient des `.*`, nosu pouvons penser que cette variable correspond à un format de fichier à reconnaître.

Expression régulière 5 : `$ligne =~ /$e/` : vérifie que la ligne du fichier (`$ligne`) contient la variable `$e`.
Pour chaque fichier correspondant au format demandé, celui-ci est ouvert, et si un certain nombre de ses lignes contient le motif recherché, ce nombre va être mémorisé dans un tableau associatif à deux dimensions nommé résultats.

Synthèse explicative :

```
Exemple d'appel : examen.pl  ~  sardine.*anchoix  *.doc

le script parcourt tous les répertoires contenus dans ~ à la recherche de fichiers (format .doc)
(ce format aura été transformé en l'expression régulière .*.doc (voir remarques à la fin)),
et compte dans chaque fichier les lignes contenant le mot "sardine" suivi plus loin d'"anchoix".

Ces informations sont regroupées, puis affichées comme ceci :
tri sur le nombre de lignes trouvées
    répertoires contenant les fichiers correspondants
       liste des noms des fichiers séparés par des :

Par exemple :
4 appariements :
    RECETTES_DE_LA_MER : recette1.doc:recette4.doc:recette5.doc
    RECETTES_AUTOUR_DES_ANCHOIX : recette8.doc
5 appariements :
    RECETTES_MERVEILLEUSES : recette21.doc
```

Bien sûr, vous aurez tristement remarqué :
- que le test sur le nombre de paramètres requis est mauvaix ;
- que les points ne sont pas déspécialisés quand le format de fichier contenant les jokers est transformé en expression régulière.

(Un étudiant de la promotion concernée l'avait vu ;-))

Ecriture d'un script CGI (9 points)

```
#!/usr/bin/perl

sub analyse_parametres {
  @parametres = split /&/, $_[0];
  foreach $parametre (@parametres) {
     $parametre =~ /(.*)=(.*)/;
     $params{$1} = $2;
  }
}
analyse_parametres($ENV{'QUERY_STRING'});
$op1 = $params{'op1'};
$op2 = $params{'op2'};
$operateur = $params{'operateur'};

if ($op1 && $operateur && $op2) {
    if ($operateur eq "plus") { $resultat = $op1 + $op2; }
    if ($operateur eq "moins") { $resultat = $op1 - $op2; }
    if ($operateur eq "multi") { $resultat = $op1 * $op2; }
    if ($operateur eq "div") { $resultat = $op1 / $op2; }
}

print <<DATA;
Content-type: text/html

<html>

   <body>
   <center>
   <h2> Calculette avec mémoire </h2>
   <form action="examen_IC_Perl_2009.cgi">

         <input name="op1" type="text" size="5" value="$resultat"></input>
         <select name="operateur">
             <option value="plus">+<option>
             <option value="moins">-<option>
             <option value="multi">*<option>
             <option value="div">/<option>
         </select>
         <input name="op2" type="text" size="5"></input>
         <br/><br/>
         <input type="submit" value="Lancez le calcul"></input>
   </form>

   </body>

</html>
DATA
```