Master Intégration de Compétences
Examen de Système

Vincent Berry et Pierre Pompidor

11 décembre 2007 - SC. 1.01 - Tout document autorisé

Questions de cours : 6 points

Paramétrage du shell (2 points)

Quels sont les paramétrages courants de l'environnement qui sont spécifiés dans le fichier **.bashrc** ?

Droits d'accès (2 points)

Dans le cadre d'un travail en projet, vous voulez créer un répertoire dans lequel seuls vos collègues auront le droit de déposer et d'accéder aux fichiers du *groupe* de travail. Comment pouvez-vous faire (spécifiez les commandes utilisées) ?

Recherche (2 points)

Vous voulez rechercher tous les fichiers HTML qui affichent une image (balise img) et qui se trouvent quelque part dans l'un de vos répertoires. Comment faire (spécifiez également les commandes utilisées) ?

Compréhension d'un script CGI : 6 points

Décrivez ce que fait le script suivant (examen_2007_IC.cgi) :
- en expliquant chacune de ses trois parties (2 points chacune) ;
- en décrivant le contenu des variables suivant un exemple que vous choisirez ;
- en dessinant un ou plusieurs dessins d'écran.

```
#!/usr/bin/perl

### Partie 1 :
@params = split /&/, $ENV{'QUERY_STRING'};
# split est une fonction qui decoupe une chaine de caracteres
# suivant le motif defini par l'expression reguliere (encadree par deux /)
foreach $parametre (@params) {
    if ($parametre =~ /(.*)=(.*)/) {
       $parametres{$1} = $2;
   }
}
$m = $parametres{'m'};
$c = $parametres{'c'};

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

<html>
  <body>
DATA

### Partie 2 :
@lm = split //, $parametres{'m'};
foreach $lettre (@lm) {
    if ($lettre eq $c) {print "$c"; } else {print "."; }
}

### Partie 3 :
print "<br/><br/>";
foreach $c ('a', 'e', 'i', 'o', 'u', 'y') {
    print "<a href=\"examen_2007_IC.cgi?m=$m&c=$c\"> $c </a> <br/>";
}

print <<DATA;
  </body>
</html>
DATA
```

Ecriture de scripts : 8 points

Une des erreur de syntaxe les plus courantes que vous rencontrez dans vos TP concerne les symboles parenthésés (parenthèses, accolades ou crochets). On se propose ici d'écrire un petit analyseur de code ayant pour but de détecter des bugs sur le placement des accolades. Les questions q2 et q3 peuvent être faites indépendamment l'une de l'autre

Q1 : Comptage d'accolades (3 points)

Ecrivez un script `accolades.pl` qui accepte en paramètre le nom d'un fichier contenant un script perl ou un programme java et qui compte le nombre d'accolades ouvrantes et fermantes dans le contenu de ce fichier.

Pour cette question, nous supposerons que le programme à analyser ne contient pas plus d'une accolade ouvrante et d'une accolade fermante par ligne de code.

Après avoir compté le nombre d'accolades ouvrantes et fermantes, votre script doit afficher un message indiquant si le bon nombre d'accolades est présent ou s'il détecte un problème.

Dans cette première question vous ne devez pas vous préoccuper de l'ordre dans lequel interviennent les accolades, même si cet ordre peut être incorrect : concentrez-vous sur le comptage des accolades. Par exemple soit le script `code.pl` suivant :

```
#!/usr/bin/env perl
$param = $#ARGV{0];
if ($param =~ /\.pl/) {
        print "c'est un script perl !";
}
print "ensuite\n";
while ($i<=10) {
        print "i vaut $i\n";
print "fin\n";
```

Si dans un terminal, on tape la commande suivante : `accolades.pl code.pl` alors le message suivant doit apparaître à l'écran :

```
ERREUR : il manque 1 accolade fermante dans code.pl
```

Q2 : Multiples occurrences sur la même ligne (2 points)

Proposez une version `accolades2.pl` du script précédent qui tienne compte du fait que les lignes du code à analyser peuvent contenir plusieurs accolades sur la même ligne (ouvrantes et/ou fermantes). Vous pourrez décrire `accolades2.pl` en indiquant uniquement ses différences avec `accolades1.pl`.

Q3 : Ordre d'apparition des accolades (3 points)

On veut maintenant une version `accolades3.pl` qui prenne en compte l'ordre dans lequel apparaisse les accolades et non plus seulement le nombre. Par exemple la séquence suivante :
`{ ... { ... } ... } ... } ... {`
contient le même nombre d'accolades fermantes et ouvrantes, mais est incorrecte : au moment où l'on rencontre la 3ème accolade fermante, seulement deux accolades ouvrantes ont été rencontrées, ce qui signifie qu'une de ces fermantes n'a pas d'ouvrante lui correspondant.

Bonus : Problème de conception (2 points)

Pouvez-vous expliquer pourquoi les scripts précédents peuvent parfois penser détecter une erreur en analysant les accolades, sans que le script analysé soit incorrect ? Proposez une piste pour corriger ce problème. Votre solution sera exprimée sous la forme d'un texte explicatif de 5 à 20 lignes, sans code.