

0.1 Module : CGI+SSI

0.1.1 1. Introduction

Voici comment modifier la configuration de votre serveur web afin d'activer les CGI et exploiter les SSI. **CGI** : Common gateway interface. Une cgi est un programme dont le flux de sortie sera renvoyé au client via le serveur web.

- **SSI** : Server side include. Les SSI sont des instructions que l'on peut inclure dans une page web, qui sera analysée par le serveur avant d'être envoyée au client.

0.1.2 2. Configuration d'apache

Commençons d'abord par "activer" les cgi, pour cela, il suffit de modifier le fichier httpd.conf, et trouvez la ligne

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
```

Le deuxième argument peut être différent, cherchez dans le fichier simplement la ligne qui contient ScriptAlias, et remplacez /var/www/cgi-bin/ par le répertoire qui contiendra les cgi, si cette ligne est manquante ajoutez-la simplement.

```
AddHandler cgi-script .pl
AddHandler cgi-script .cgi
AddHandler cgi-script .exe
```

Ces lignes permettent au serveur de savoir que les fichiers ayant des extensions en .pl, .cgi et .exe sont des CGI à exécuter. Ensuite faites un petit test, tapez ceci dans un fichier que vous nommez par exemple test.cgi

```
#!/bin/sh
# disable filename globbing
set -f
echo Content-type : text/html
echo
echo "<b>Les cgi ont l'air de fonctionner</b>"
echo
```

Vous devez également systématiquement attribuer les droits nécessaires à votre fichier pour qu'il puisse être exécuté, pour cela tapez simplement :

```
chmod a+x /votre/site/cgi-bin/test.cgi
```

Ensuite testez via votre navigateur en tapant dans la barre d'adresse : <http://localhost/cgi-bin/test.cgi>¹ Et vous devriez voir apparaître un texte en gras. Maintenant nous allons "activer" les ssi dans un répertoire. Pour cela, vérifiez que la ligne

```
LoadModule includes_module libexec/mod_include.so
```

est bien décommentée. Ensuite ajoutez (toujours dans httpd.conf) les lignes :

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Et ensuite pour chaque répertoire qui contiendront des fichiers contenant des SSI ajoutez :

¹<http://localhost/cgi-bin/test.cgi>

```
<Directory /votre/site/>
Options Indexes Includes FollowSymLinks MultiViews
</Directory>
```

Vous pouvez simplement rajouter cette ligne à la fin du fichier. Supposons que vous ayez activé les SSI dans le root du serveur web. Testons le bon fonctionnement des SSI en créant par exemple ce fichier :

```
<html>
<head><title>test SSI</title></head>
<body>
<div><!--#echo var="REMOTE_ADDR" --></div>
<div><!--#exec cgi="cgi-bin/test.cgi" --></div>
</body>
</html>
```

Que vous enregistrez sous test.shtml (attention à l'extension). Si vous l'avez placé à la racine du serveur Vous devriez pouvoir l'atteindre en tapant `http://localhost/test.shtml` dans votre navigateur.

0.1.3 3. Ecrire des CGI

Les cgi peuvent être écrites dans divers langages, les plus courants sont perl,python,C/C++ mais des tas d'autre langage permettent d'implémenter des CGI. Le concept est toujours le même, une cgi emploie le flux de donnée standart ou l'on spécifie une entete relative au contenu généré par votre CGI. Pour du html il s'agit simplement de "Content-type : text/html". Vous devez donc effectuer une sortie de cette constante avant toutes autres sorties. Vous disposez ci-dessus d'un exemple en bash. Voici un exemple en C++ et perl :

0.1.4 3.1 en perl :

une CGI de base en Perl serait :

```
#!/usr/bin/perl
print "Content-type : text/html\n\n";
print "<div>Hello world!!!</div>";
```

Tout simplement. Enregistrez ce code dans un fichier que vous nommez par exemple test.pl dans le repertoire cgi-bin n'oubliez pas la commande `chmod a+x /votre/site/cgi-bin/test.pl` Pour exploiter les variables d'environnement de apache, vous pouvez utiliser le module CGI.pm soit la variable global \$ENV, modifiez test.pl de cette façon :

```
#!/usr/bin/perl
use CGI;
$co = new CGI;
$test = $co->param('test');
#on recupere les variables postées ou transmises par l'url
print "Content-type : text/html\n\n";
print "<div><b>test = ".$test."</b></div>";
print "<div>votre ip :".$ENV{'REMOTE_ADDR'}."</div>";
#on affiche l'adresse ip sans le module CGI
```

Pour comprendre au mieux ce dernier script, tapez par exemple `http://localhost:80/cgi-bin/test.pl?test=foo`² dans votre navigateur, de sorte que perl puisse assigner une valeur à `$test` lors de l'instruction `$test = $co->param('test');`;

0.1.5 3.2 en cpp :

Le "hello world" typique :

```
#include <iostream.h>
int main()
{
  cout<<"Content-type : text/html\n\n"«endl;
  cout<<"<b>Hello world!!!</b>"«endl;
  return 0;
}
```

enregistrer ceci dans un fichier `test.cpp` par exemple ensuite taper la commande

```
g++ test.cpp -o testcpp.cgi
```

copier `testcpp.cgi` dans votre repertoire `cgi-bin` (n'oubliez pas la commande `chmod`), et tapez dans votre navigateur `http://localhost/cgi-bin/testcpp.cgi`³ Pour exploiter les variables d'environnement, il suffit d'inclure `stdlib.h` pour disposer de la fonction `getenv`, modifier `testcpp.cpp` :

```
#include <iostream.h>
#include <stdlib.h>
int main()
{
  char * ip = getenv("REMOTE_ADDR");
  cout<<"Content-type : text/html\n\n"«endl;
  cout<<"<b>Hello world!!!</b>"«endl;
  cout<<"<div>votre ip : "<ip>"</div>"«endl;
  return 0;
}
```

recompilez et recopiez `testcpp.cgi` au bon endroit et réexécutez le programme via votre navigateur vous devriez voir "hello world" suivi de votre adresse ip. Les variables postées peuvent également être lu sur le flux d'entrée standard avec la fonction `cin` par exemple : Créer d'abord un fichier `form.html` que vous ne placez pas dans votre repertoire `cgi-bin` pour éviter tout problème de droit, par exemple à la racine : `form.html` :

```
<html>
<form action = "cgi-bin/testcpp.cgi" method="post">
<input type="text" name="test" />
<input type="submit" value="envoyer" />
</form>
</html>
```

²<http://localhost:80/cgi-bin/test.pl?test=foo>

³<http://localhost/cgi-bin/testcpp.cgi>

et modifier test.cpp de cette façon :

```
#include <iostream.h>
#include <stdlib.h>
int main()
{
char * ip = getenv("REMOTE_ADDR");
int length = atoi(getenv("CONTENT_LENGTH"));
char i[10];
cout<<"Content-type : text/html\n\n"<<endl;
cout<<"<div>hello world</div>";
cout<<"<div>";
for (int j = 0; j< length; j++)
cin >> i[j];
for (int j = 0; j< length; j++)
cout<< i[j];
cout<<"</div>";
return 0;
}
```

Vous devriez voir "hello world" puis un retour a la ligne puis encore "test=" suivi de ce que vous avez entré dans le formulaire.

0.1.6 4. Variable d'environnement :

Voici une liste non-exhaustive des variables d'environnement d'apache : (que vous pouvez passer en arguments aux méthodes getenv et \$ENV des codes ci-dessus)

- \$ENV{'CONTENT_LENGTH'} taille des données, peut servir a déterminer la taille des données envoyées lors d'une requete http, peut determiner par exemple la taille d'un fichier uploader
- \$ENV{'CONTENT_TYPE'} type des données , peut servir également dans le cas d'un upload, si on veut s'assurer que le client uploade un fichier de tel type ex : html,jpg,etc...
- \$ENV{'HTTP_COOKIE'} renvoie le contenu des cookies du client
- \$ENV{'HTTP_REFERER'} renvoie l'url d'ou vient le client.
- \$ENV{'HTTP_REQUEST_METHOD'} renvoie le type de la méthode post ou get par exemple.
- \$ENV{'HTTP_USER_AGENT'} renvoie le nom du navigateur du client ...
- \$ENV{'QUERY_STRING'} contient les données de l'url.
- \$ENV{'REMOTE_ADDR'} contient l'adresse ip du client

Remarques :

- vous avez sans doute remarqué que la fonction getenv en c/c++ renvoie les données sous forme : "var1=ttt&var2=hhhh" ,il faut donc analyser la chaîne pour en extraire les données.En cherchant sur le web vous trouverez très facilement des classes toutes faites, pour vous aider a exploiter les CGI en c++.