Le serveur Web : Apache

Stéphane Gill

Stephane.Gill@CollegeAhuntsic.qc.ca

Table des matières

Introduction

Un serveur Web est un logiciel, dont le rôle est d'écouter des requêtes d'un type particulier (requête http) provenant d'un client que l'on appelle navigateur. Les navigateurs sont bien connus des utilisateurs, et les plus populaires sont Internet Explorer, Mozilla, FireFox et Opera.

Dans ce document, une brève présentation du protocole http et du langage html est présenté, avant de s'intéresser à l'installation et la configuration d'un serveur LAMP (Linux, Apache, MySQL et php).

Protocole http

Le protocole HTTP (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990. La version 0.9 était uniquement destinée à transférer des données sur Internet (en particulier des pages Web écrites en HTML. La version 1.0 du protocole (la plus utilisée) permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME.

Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement des fichiers HTML) localisé grâce à une chaîne de caractères appelée URL entre un navigateur et un serveur Web

Communication entre navigateur et serveur

La communication entre le navigateur et le serveur se fait en deux temps :

- Le navigateur effectue une requête HTTP
- Le serveur traite la requête puis envoie une réponse HTTP

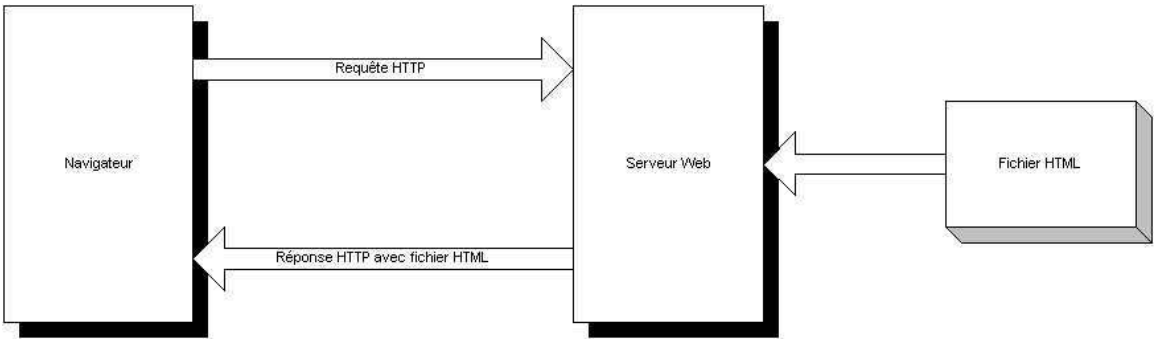

Figure 1 : protocole HTTP

En réalité la communication s'effectue en plus de temps si on considère le traitement de la requête par le serveur. Étant donné que l'on s'intéresse uniquement au protocole HTTP, le traitement du côté serveur ne sera pas explicité.

Requête HTTP

Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur. Elle comprend:

- **une ligne de requête :** c'est une ligne précisant le type de document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. La ligne comprend trois éléments devant être séparé par un espace:
  - La méthode
  - L'URL
  - La version du protocole utilisé par le client (généralement HTTP/1.0)
- **Les champs d'en-tête de la requête :** ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (Navigateur,système d'exploitation,...). Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la requête :** ensemble de ligne optionnelle devant être séparé des lignes précédentes par une ligne vide et permettant par exemple un envoi de données par une commande POST lors de l'envoi de données au serveur par un formulaire

Voici donc un exemple de requête HTTP:

```
GET http://www.commentcamarche.net HTTP/1.0
Accept : text/html
If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

Réponse HTTP

Une réponse HTTP est un ensemble de lignes envoyé au navigateur par le serveur. Elle comprend:

- **une ligne de statut :** c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparé par un espace:
  - La version du protocole utilisé
  - Le code de statut
  - La signification du code
- **Les champs d'en-tête de la réponse :** ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composé d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la réponse :** Il contient le document demandé

Voici donc un exemple de réponse HTTP:

```
HTTP/1.0 200 OK
Date : Sat, 15 Jan 2000 14:37:12 GMT
Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Lentgh : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```


Introduction à HTML

Le langage HTML tire son origine du langage SGML (Standard Generalized Markup Language). Il s'agit d'un type particulier d'annotations destiné au WWW et qui correspond à une collection de styles reconnaissables par les navigateurs. Un navigateur est donc un logiciel qui interprète à l'écran les commandes HTML contenues dans un document accessible sur le WWW.

Le langage HTML est utilisé sur le WWW depuis 1990. La version actuellement en vigueur est HTML 4.0. Dans cette section, une brève description du langage HTML est présentée.

Pages HTML

Il faut d'abord spécifier qu'un document contenant des annotations en HTML n'est rien de plus qu'un fichier texte. Il peut donc être reconnu sans problèmes de conversion d'un environnement à un autre. Une page peut donc être lue et interprétée par n'importe quel navigateur sur n'importe quelle plate-forme.

Pour créer un document HTML, il faut insérer les commandes suivantes au début du document:

```
<HTML> "première ligne du document"
<head> "ouverture de la zone d'en-tête"
<title> "titre de la page suivi de </title>"
</head> "fermeture de la zone d'en-tête."
<body> "ouverture du corps du document"
"Mettre le texte et les images ici"
</body> "fin du corps du document"
</HTML> "fin du document HTML"
```

Exemple :

```
<HTML>
<HEAD>
<TITLE>HyperText Test</Title>
</HEAD>
<Body>
<H1>Bonjour a tous</H1>
<P>La vie est belle.<\P>
</Body>
</HTML>
```

Formatage du texte

Cette section décrit les différentes options disponibles pour formater le texte dans une page WWW.

Entêtes (Headers)

Le formatage d'une page commence généralement par le choix et l'usage d'en-têtes prédéterminées qui s'échelonnent de H1 à H6 (niveaux). La commande H1 est la plus grosse disponible et la H6 est la plus petite. Les commandes Hx comprennent un choix de taille, le caractère gras et un retour de paragraphe. Par exemple <H1> suivi de l'entête et de </H1> donne:

Exemple de header de niveau 1

Il n'est pas nécessaire d'ajouter les commandes <P> ou
 à la fin des lignes de H1 à H6 puisque celles-ci sont implicites.

Autres commandes

Commande	Description
texte	Centrer le texte
<P>	changement de paragraphe
 	changement de ligne
texte	texte en gras
	texte en gras
<EM></EM>	texte en italique
<I></I>	texte en italique
<CITE></CITE>	texte en italique
<TT></TT>	texte formaté avec une fonte à espacement constant

note:

- Les majuscules et les minuscules peuvent être utilisées indifféremment pour les annotations.

- Les commandes <P> et
 n'ont pas à être fermées. Vous pouvez utiliser plusieurs commandes
 ou <P> répétitivement pour augmenter l'espacement.

Caractères spéciaux dans HTML

Certains caractères ont une signification spécifique dans HTML. Pour les utiliser comme tels dans une page, il faut utiliser les commandes alternatives pour les afficher correctement à l'écran. Ces commandes sont:

< pour: **<**

> pour: **>**

& pour: **&**

" pour: **"**

Images

Les navigateurs HTML reconnaissent généralement deux formats d'images; les images GIF et les images JPEG. Ces deux formats d'images sont comprimés. L'insertion d'une image est possible en tapant la commande suivante: <img src="images/serviette50.gif">, ce qui donne:

Le segment `IMG SRC` indique qu'il s'agit de la source d'une image, le premier terme entre les guillemets indique le nom du dossier où se trouve la ou les images, le deuxième terme indique le nom du fichier contenant l'image et le format de celle-ci. Une image en format GIF se termine par `.GIF` alors qu'une image en format JPEG se termine par `.JPEG ou .JPG` .

La taille de l'image est déterminée par le fichier lui-même. On peut placer par exemple une grande image occupant l'ensemble de l'écran:

Lignes horizontales

Pour créer des lignes en HTML, la commande de base est `<HR>`. Pour faire varier la largeur de la ligne, on utilise le code suivant: `<HR WIDTH=75%>` . Cela donne une ligne de 75% de la largeur

de l'écran (selon la largeur de la fenêtre choisie par l'usager). Pour faire varier l'épaisseur de la ligne: <HR SIZE="5"> donne une ligne d'épaisseur 5:

Les lignes peuvent aussi être alignées à gauche à droite ou centrées, on utilise alors les commandes suivantes :

```
<HR ALIGN=left>
<HR ALIGN=right>
<HR ALIGN=center>
```

Pointeurs

Pour insérer un pointeur (lien hypertexte-hypermédia), il faut indiquer une référence (appelée URL pour Uniform Ressource Locator) et un élément, texte ou image, visible à l'écran sur lequel on doit cliquer pour y accéder. Voici un exemple de code pour obtenir un pointeur:

```
<A HREF="dossier/menu_du_jour.HTML">Menu du jour</A>.
```

Le pointeur apparaît alors en couleur contrastée (bleu par défaut dans Netscape) et souligné dans le navigateur. Les pointeurs peuvent diriger le navigateur vers des sites HTTP, FTP, TELNET, TN3270, GOPHER ou USENET. On peut aussi entrer directement sur un fichier sur le système local ou sur un réseau local. Pour créer un pointeur, il s'agit tout simplement de définir le type de document dans la commande A HREF comme dans les exemples qui suivent.

Site HTTP (WWW)

Le code:

```
<A HREF="http://WWW.fsaa.ulaval.ca">Serveur WWW de la FSAA</A>
```

donne accès à un serveur WWW, notamment à celui de la Faculté des sciences de l'agriculture et de l'alimentation de l'Université Laval.

Fichier sur le système hôte

Le code:

```
<A HREF="fichier.HTML">fichier</A>
```

donne tout simplement accès à un fichier HTML situé au même niveau hiérarchique que le fichier actuellement ouvert sur le serveur.

Installation d'un serveur LAMP

L'objet de cette section est de guider l'administrateur débutant dans l'installation de la trilogie Apache, PHP, MySQL. Depuis un certain nombre de versions de Red Hat Linux, il est possible de tout faire fonctionner sans retoucher le moindre fichier de configuration, simplement en installant les rpm.

Rappelons brièvement que:

- **Apache** est un serveur Web. I s'agit du serveur le plus utilisé actuellement sur le Web puisqu'il représente plus de 70% des serveurs installés.
- **PHP** est un langage de programmation interprété. Correctement interfacé avec Apache il permet au serveur de fournir des pages dynamiques gérées en fonction des besoins du client. En clair, la page n'est plus un document statique mais peut évoluer, afficher des informations différentes selon les souhaits de l'utilisateur.
- **MySQL** est un gestionnaire de bases de données. Il peut très bien fonctionner en utilisant son propre client en mode texte sans l'utilisation d'une quelconque interface graphique.

Installation d'Apache

L'installation d'Apache sous Fedora Core 1 ne pose habituellement pas de problèmes, il suffit d'installer les paquetages suivants :

- httpd-devel-2.0
- httpd-2.0

Le fonctionnement d'Apache est matérialisé par la mise en route d'un démon nommé httpd. Pour mettre en route Apache, il suffit d'exécuter la commande:

```
service httpd start
```

Il est possible de passer immédiatement au test du serveur sans passer par la moindre phase de configuration. Pour cela un navigateur Web doit être démarrer puis l'URL suivant doit être saisie :

```
http://127.0.0.1/
```


La page de présentation de Apache devrait apparaître. Par défaut, Apache est configuré de façon tel que les pages Web doivent être installés dans le répertoire */var/www/html*.

Installation de PHP

L'installation de PHP se fait en utilisant les paquetages suivants :

- php-4.3
- php-devel-4.3

Une fois l'installation des paquetages rpm terminé, Apache doit être redémarré afin de prendre en compte le module PHP.

Il importe maintenant de vérifier le fonctionnement de PHP en créant un script *test.php* dans le répertoire */var/www/html/* :

```
<html>
<?
echo 'premier test php<br><br>';
phpinfo();
?>
</html>
```

À partir d'un navigateur l'URL *http://127.0.0.1/test/test.php* permet de charger le script.

Installation de MySQL

L'installation de mysql est elle aussi très simple. Les packages suivants sont nécessaires (ils devraient avoir été installés par défaut, en cas de besoin les installer par la suite):

- mysql-3.23.58-4
- mysql-server-3.23.58-4
- mysql-devel-3.23.58-4
- php-mysql-4.3

Le démarrage du serveur MySQL s'effectue avec la commande suivante :

```
service mysql start
```


L'installation par défaut utilise root comme compte administrateur du serveur et aucun mot de passe. Le fonctionnement du serveur MySQL peut être testé en utilisant simplement le client MySQL texte.

```
# mysql -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.18-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Pour quitter MySQL, il suffit de taper *quit* à l'invite MySQL.

Installation de phpMyAdmin

Plutôt que de rédiger un script php qui va s'adresser au serveur MySQL il est préférable d'installer un ensemble de scripts nommés phpMyAdmin. Il s'agit tout simplement d'un ensemble de scripts PHP qui permet, via un navigateur Web, d'administrer des bases de données sous MySQL. Les sources de phpMyAdmin sont disponible à cette adresse http://www.phpmyadmin.net/.

Fichier de configuration httpd.conf

Paramètre de configuration globale

```
### Section 1: Global Environment
ServerTokens OS
ServerRoot "/etc/httpd"
#ScoreBoardFile run/httpd.scoreboard
PidFile run/httpd.pid
Timeout 300
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
```

Servertokens

Le paramètre Servertokens spécifie la chaîne d'identification du serveur. On a le choix entre Prod, Min, OS, Full.

- Prod[uctOnly] : Server: Apache
- Min[imal] : Server: Apache/1.3.0
- OS : Server: Apache/1.3.0 (Unix)
- Full (or not specified) : Server: Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2

ServerRoot

Le répertoire ServerRoot est le répertoire de niveau supérieur contenant les fichiers du serveur. Tant le serveur sécurisé que le serveur non-sécurisé établissent la directive ServerRoot à "/etc/httpd".

ScoreBoardFile

ScoreBoardFile stocke les informations internes au processus serveur utilisées pour la communication entre le processus serveur parent et ses processus enfants. Red Hat Linux utilise la mémoire partagée pour stocker ScoreBoardFile, la valeur par défaut /etc/httpd/logs/apache_runtime_status n'est utilisée qu'en cas de secours.

PidFile

PidFile est le nom du fichier dans lequel le serveur consigne son identifiant de processus (PID). Le PID par défaut est /var/run/httpd.pid.

Timeout

Timeout définit, en secondes, la durée pendant laquelle le serveur attend des réceptions et des émissions en cours de communication. Plus spécifiquement, Timeout définit la durée pendant laquelle le serveur attend de recevoir une requête GET, la durée pendant laquelle il attend de recevoir des paquets TCP sur une requête POST ou PUT et la durée pendant laquelle il attend entre des ACK répondant aux paquets TCP. La valeur de Timeout est réglée à 300 secondes par défaut, ce qui est approprié dans la plupart des cas.

KeepAlive

KeepAlive définit si votre serveur autorisera plus d'un requête par connexion; cette directive peut servir à empêcher un client particulier d'utiliser une trop grande quantité des ressources du serveur.

Par défaut, la valeur de Keepalive est réglée sur off. Si la valeur de Keepalive est on et que le serveur devient très occupé, le serveur peut générer rapidement un maximum de processus enfants. Dans ce cas, les serveur sera considérablement ralenti. Si la directive Keepalive est activée, il est recommandé de donner à KeepAliveTimeout une valeur basse

MaxKeepAliveRequests

Cette directive définit le nombre maximum de requêtes autorisées par connexion persistante. Le groupe Apache Project recommande l'utilisation d'un paramétrage élevé, ce qui améliorera les performances du serveur. Par défaut, la valeur de MaxKeepAliveRequests est paramétrée sur 100, ce qui est approprié pour la plupart des situations.

KeepAliveTimeout

KeepAliveTimeout définit la durée en secondes pendant laquelle votre serveur attendra, après avoir servi une requête, avant d'interrompre la connexion. Une fois que le serveur reçoit une requête, c'est la directive Timeout qui s'applique à sa place. Par défaut, la valeur donnée à KeepAliveTimeout est 15 secondes.

Les modules

```
##
## Server-Pool Size Regulation (MPM specific)
##
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
MaxClients       150
MaxRequestsPerChild  1000
</IfModule>
<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>

LoadModule access_module modules/mod_access.so
.  .  .
.  .  .
.  .  .
LoadModule cgi_module modules/mod_cgi.so
```

IfModule

Les balises <IfModule> et </IfModule> créent un conteneur conditionnel dont les directives ne sont activés que si le module spécifié est chargé. Les directives placées entre les balises IfModule sont traitées dans l'un des deux cas suivants. Les directives sont traitées si le module contenu dans la balise de début <IfModule> est chargé. Ou, si un point d'exclamation ([!]) figure devant le nom du module, les directives ne sont traitées que si le module dans la balise <IfModule> n'est pas chargé.

LoadModule

LoadModule est utilisée pour charger des modules DSO (de l'anglais 'Dynamic Shared Object', objet partagé dynamique).

Configuration du serveur principal

```
### Section 2: 'Main' server configuration
#Listen 12.34.56.78:80
Listen 80

User apache
Group apache
ServerAdmin root@localhost
#ServerName new.host.name:80
UseCanonicalName Off
DocumentRoot "/var/www/html"
```

Listen

The Listen identifie les ports sur lesquels votre serveur Web acceptera les demandes entrantes. Par défaut, le Serveur HTTP Apache est paramétré pour écouter sur le port 80 pour les communications Web non sécurisées et (dans /etc/httpd/conf.d/ssl.conf définissant tout serveur sécurisé) sur le port 443 pour les communications Web sécurisées.

Si le Serveur HTTP Apache est configuré pour écouter sur un port dont le numéro est inférieur à 1024, il doit être lancé en tant que super-utilisateur. En revanche, pour les ports dont le numéro est égal ou supérieur à 1024, httpd peut être lancé en tant que simple utilisateur.

La directive Listen peut également être utilisée pour spécifier des adresses IP particulières sur lesquelles le serveur acceptera des connexions.

User

La directive User définit le nom d'utilisateur du processus serveur et détermine les fichiers auxquels le serveur peut avoir accès. Tous les fichiers inaccessibles à cet utilisateur seront également inaccessibles aux clients se connectant au Serveur HTTP Apache.

La valeur par défaut donnée à User est apache.

Remarque : Pour des raisons de sécurité, le Serveur HTTP Apache refuse d'être exécuté en tant que super-utilisateur (ou root). user.

Group

Spécifie le nom de groupe des processus du Serveur HTTP Apache.

La valeur par défaut attribuée à Group est apache.

ServerAdmin

Donnez comme valeur à la directive ServerAdmin l'adresse électronique de l'administrateur du serveur Web. Cette adresse électronique apparaîtra dans les messages d'erreur sur les pages Web générées par le serveur afin que les utilisateurs puissent signaler un problème en envoyant un message électronique à l'administrateur du serveur.

La valeur par défaut donnée à ServerAdmin est root@localhost.

Généralement, la valeur donnée à ServerAdmin est Webmaster@example.com. Vous pouvez ensuite créer un alias pour Webmaster au nom de la personne responsable du serveur Web dans /etc/aliases et exécuter /usr/bin/newaliases.

ServerName

Utilisez ServerName pour définir un nom d'hôte et un numéro de port (en accord avec la directive Listen) pour le serveur. La directive ServerName ne doit pas forcément correspondre au nom d'hôte de l'ordinateur. Par exemple, le serveur Web pourrait être www.example.com bien que le nom d'hôte de l'ordinateur soit foo.example.com. La valeur spécifiée dans ServerName doit être un nom de domaine (ou DNS, de l'anglais 'Domain Name Service') valide qui peut être résolu par le système — ne vous contentez surtout pas d'en inventer un.

Ci-dessous figure un exemple de directive ServerName:


```
ServerName www.example.com:80
```

Lors de la détermination d'un ServerName, assurez-vous que son adresse IP et son nom de serveur sont bien inclus dans le fichier /etc/hosts.

UseCanonicalName

Lorsque la valeur attribuée à cette directive est on, elle configure le Serveur HTTP Apache de manière à ce qu'il se référence en utilisant les valeurs précisées dans les directives ServerName et Port. En revanche, lorsque la valeur de UseCanonicalName est off, le serveur emploiera la valeur utilisée le client envoyant la requête lorsqu'il fait référence à lui-même.

Par défaut, la valeur attribuée à UseCanonicalName est off.

DocumentRoot

DocumentRoot est le répertoire contenant la plupart des fichiers HTML qui seront servis en réponse aux requêtes. La valeur par défaut pour DocumentRoot aussi bien pour le serveur Web sécurisé que pour se serveur Web non-sécurisé est le répertoire /var/www/html. Par exemple, le serveur pourrait recevoir une demande pour le document suivant:

```
http://example.com/foo.html
```

Le serveur recherche le fichier suivant dans le répertoire par défaut:

```
/var/www/html/foo.html
```

Directives de blocs

```
<
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Directory

<Directory> et </Directory> sont utilisés pour "encapsuler" un groupe de directives applicables uniquement au réprtoire indiqué ainsi qu'à ses sous-répertoires. Toute directive autorisée dans un contexte de répertoire peut apparaître entre ces deux balises. nomrépertoire est soit le chemin entièrement qualifié du répertoire, ou un motif. Dans un motif, '?' remplace un caractère unique quelconque, et '*' remplace toute séquence de zéro ou plus caractères quelconques. Sur Apache 1.3, vous pouvez aussi utiliser les plages de caractères '[]' comme dans un shell UNIX. De plus aucun des métacaractères ne peut remplacer un '/', ce qui correspond plus intimement à la réaction des shells UNIX. Exemple:

```
   <Directory /usr/local/httpd/htdocs>
   Options Indexes FollowSymLinks
   </Directory>
```

A partir d'Apache 1.2 : Peuvent être utilisées les "expressions régulières", lesquelles devront être précédées du caractère ~. Par exemple :

```
<Directory ~ "^/www/.*/[0-9]{3}">
```

correspondrait à des répertoires dans /www/ dont le nom serait constitué de trois digits.

Si plusieurs sections de répertoires pointent sur le répertoire d'un document (ou l'un de ses pères) sans qu'il s'agisse d'une expression régulière, alors les directives sont appliquées selon la loi de "la plus courte qualification d'abord", combinées aux directives des fichiers .htaccess. Par exemple, avec l'écriture

```
<Directory />
AllowOverride None
</Directory>
```

```
<Directory /home/*>
AllowOverride FileInfo
</Directory>
```

pour le contrôle d'accès au document /home/web/dir/doc.html les étapes d'évaluation sont les suivantes :

Applique la directive AllowOverride None (désactivant les fichiers .htaccess).

Appliquela directive AllowOverride FileInfo (pour le répertoire /home/web).

Applique toutes les directives FileInfo de /home/web/.htaccess

Les sections exprimant des répertoires sous forme d'expressions régulières sont gérés légèrement différemment par Apache 1.2 et 1.3. Sous Apache 1.2, elles sont combinées aux sections "normales" et s'appliquent dans l'ordre où elles apparaissent dans le fichier de configuration. Elles ne s'appliquent qu'une fois, seulement pour celles qui font partie de la section "à plus courte correspondance". Sous Apache 1.3 les sections basées sur des expressions régulières ne sont pas évaluées tant que toutes les sections "normales" n'ont pas été considérées. A ce moment, les sections "régulières" sont traitées dans l'ordre où elles apparaissent dans le fichier de configuration. Par exemple, avec l'écriture

```
<Directory ~ abc$>
... directives ici ...
</Directory>
```

Supposez que le nom de fichier demandé soit /home/abc/public_html/abc/index.html. Le serveur considère chacune des sections /, /home, /home/abc, /home/abc/public_html, et /home/abc/public_html/abc dans cet ordre. Sous Apache 1.2, lorsque /home/abc est pris en compte, l'expression régulière correspondra et ses termes seront appliqués. Sous Apache 1.3 l'expression régulière n'est pas considérée du tout à ce point de l'arbre. Elle ne le sera pas tant que toutes les sections "normales" <Directory>s et celles des fichiers .htaccess n'ont pas été appliquées. A ce moment seulement l'expression régulière reconnaîtra /home/abc/public_html/abc et les directives seront appliquées.

Notez que l'accès par défaut d'Apache pour les sections <Directory> est Allow from All. Ceci veut dire que par défaut, Apache desservira tout fichier indiqué par une URL. Nous recommandons de modifier ceci à l'aide d'un bloc tel que

```
<Directory />
     Order Deny,Allow
     Deny from All
</Directory>
```

puis désactiver sélectivement la protection pour les répertoires devant rester accessibles. Voir la page Security Tips pour plus de détails.

Les sections de répertoires apparaissent habituellement dans le fichier access.conf, mais peuvent être présentes dans n'importe quel fichier de configuration. Les directives <Directory> ne peuvent être imbriquées, et ne peuvent petre incluses dans des sections <Limit>.

Options

La directive Options contrôle les fonctions du serveur disponibles dans un répertoire particulier. Par exemple, en vertu des paramètres restrictifs spécifiés pour le répertoire root, Options est définie uniquement sur FollowSymLinks. Aucune fonction n'est activée, à l'exception du fait que le serveur est autorisé à suivre les liens symboliques dans le répertoire root.

Par défaut, dans le répertoire DocumentRoot, Options est paramétrée pour inclure Indexes et FollowSymLinks. Indexes permet au serveur de générer le contenu d'un répertoire si aucun DirectoryIndex (par exemple, index.html) n'est spécifié. FollowSymLinks permet au serveur de suivre des liens symboliques dans ce répertoire.

- All : toutes options sauf MultiViews.
- ExecCGI : L'exécution des scripts CGI est autorisée.
- FollowSymLinks : Le serveur est autorisé à suivre les liens symboliques dans ce répertoire.
- Includes : Les inclusions par Server-Side-Include sont permises.
- IncludesNOEXEC : Les SSI sont autorisés, mais pas la commande #exec ni #include des scripts CGI.
- Indexes :Si une URL requise pointe sur un répertoire, et aucun fichier défini par DirectoryIndex (ex. index.html) n'existe dans ce répertoire, alors le serveur retourne une liste formatée du contenu du répertoire.
- MultiViews :Un contenu négocié en MultiViews est permis.
- SymLinksIfOwnerMatch :Le serveur ne suivra les liens symboliques uniquement si le fichier visé ou le répertoire visé appartiennent au même utilisateur que le lien lui-même.

AllowOverride

La directive AllowOverride définit si des Options peuvent être invalidées par les instructions d'un fichier .htaccess. Par défaut, tant le répertoire super-utilisateur que DocumentRoot sont réglés pour interdire les invalidations .htaccess.

- **AuthConfig** : Autorise l'usage de la directive Authorization (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, require, etc.).
- **FileInfo** : Autorise l'usage de directives controllant l'accès aux types de documents (AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority, etc.).
- **Indexes** : Autorise l'usage de directives controllant l'indexation des répertoires (AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName, etc.).
- **Limit** : Autorise l'usage de directives controllant les accès de certains hôtes (allow, deny et order).
- **Options** : Autorise l'usage de directives contrôlant certaines fonctionnalités spécifiques des répertoires (Options and XBitHack).

Order

La directive Order contrôle simplement l'ordre dans lequel les directives allow et deny sont analysées. Le serveur est configuré pour analyser les directives Allow avant d'analyser les directives Deny pour votre répertoire DocumentRoot.

Allow

Allow spécifie le demandeur pouvant accéder à un répertoire donné. Le demandeur peut être all, un nom de domaine, une adresse IP, une adresse IP partielle, une paire réseau/masque réseau, etc. Le répertoire DocumentRoot est configuré pour permettre (Allow) les requêtes de quiconque (all), ainsi tout le monde peut y accéder.

Deny

Deny fonctionne selon le même principe que Allow, sauf que cette fois-ci, l'accès est refusé à un demandeur donné. Le DocumentRoot n'est pas configuré par défaut pour refuser (Deny) des requêtes provenant d'un demandeur quelconque.

Site Web pour les utilisateurs

Les utilisateurs du système peuvent également mettre leur page Web à disposition du monde. Par exemple, pour l'utilisateur toto, il suffit qu'il crée un répertoire public_html/ dans son home avec un fichier index.html dedans. La page web de toto est désormais disponible au monde entier à l'adresse http://nom_DNS_de_la_machine/~toto/

Activer les sites web pour les utilisateurs

```
<IfModule mod_userdir.c>
    #UserDir disable
    UserDir public_html
</IfModule>
```

UserDir

UserDir est le nom du sous-répertoire, au sein du répertoire personnel de chaque utilisateur, où devraient être placés les fichiers HTML personnels devant être servis par le serveur Web. Par défaut, la valeur attribuée à cette directive est disable (désactiver).

Dans le fichier de configuration par défaut, le nom du sous-répertoire est public_html. Par exemple, le serveur pourrait recevoir la requête suivante:

```
http://example.com/~nom-d'utilisateur/foo.html
```

Le serveur rechercherait le fichier:

```
/home/username/public_html/foo.html
```

Dans l'exemple ci-dessus, /home/username/ est le répertoire personnel de l'utilisateur (notez que le chemin d'accès par défaut aux répertoires personnels des utilisateurs peut être différent sur votre système).

Assurez-vous que les autorisations relatives aux répertoires personnels des utilisateurs sont correctement définies. Les répertoires personnels des utilisateurs doivent être définis sur 0711. Les bits de lecture (r) et d'exécution (x) doivent être définis sur les répertoires public_html des utilisateurs (0755 fonctionnera). Les fichiers qui seront servis dans les répertoires public_html des utilisateurs doivent être définis sur au moins 0644.

Définir le répertoire public_html

```
<Directory /home/*/public_html>
     AllowOverride FileInfo AuthConfig Limit
     Options MultiViews Indexes SymLinksIfOwnerMatch
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Limit

<Limit> et </Limit> sont utilisées pour "encapsuler" un groupe de directives de contrôle d'accès ne s'appliquant uniquement qu'aux méthodes d'accès HTTP spécifiées par méthode. Toute directive autre qu'une autre directive <Limit> ou une directive <Directory> peut être utilisée; la majorité de ces directives ne seront pas affectées par <Limit>. Exemple:

```
<Limit GET POST>
require valid-user
</Limit>
```

Si une directive de contrôle d'accès apparaît en dehors d'une section <Limit>, alors elle s'applique à toutes les méthodes d'accès. Les méthodes possibles peuvent être l'une des suivantes : GET, POST, PUT, DELETE, CONNECT ou OPTIONS. Si la méthode GET est mentionnée, la restriction s'appliquera aussi aux requêtes HEAD. Si vous souhaitez implémenter une restriction pour toutes les méthodes, n'incluez aucune section <Limit>.

LimitExcept

<LimitExcept> and </LimitExcept> are used to enclose a group of access control directives which will then apply to any HTTP access method not listed in the arguments; i.e., it is the opposite of a <Limit> section and can be used to control both standard and nonstandard/unrecognized methods. See the documentation for <Limit> for more details.

For example:

```
    <LimitExcept POST GET>
    Require valid-user
    </LimitExcept>
```

Autoriser les CGI

Les CGI sont autorisés dans le répertoire */home/*/public_html/cgi-bin*.

```
<Directory /home/*/public_html/cgi-bin>
     Options ExecCGI
     SetHandler cgi-script
</Directory>
```

SetHandler

Lorsqu'elle est inscrite dans un fichier .htaccess ou à l'intérieur de sections <Directory> ou <Location>, cette directive force tous les fichiers qui correspondent à la description à être servis après traitement par le gestionnaire nommé nom-handler. Par exemple, si un répertoire ne contient que des fichiers de règles "imagemap", et pour lesquelles l'extension n'est pas significative, vous pourriez écrire ceci dans le fichier .htaccess de ce répertoire :

```
SetHandler imap-file
```

AddHandler

La directive AddHandler associe aux fichiers d'extension extension, le gestionnaire nom-handler. Par exemple, pour activer des scripts CGI écrits dans des fichiers d'extension ".cgi", vous écrirez :

```
    AddHandler cgi-script cgi
```

Une fois que cette ligne a été écrite dans votre fichier srm.conf ou httpd.conf, tout fichier dont le nom termine par ".cgi" sera considéré comme un programme CGI.

Qu'est-ce qu'un hôte virtuel ?

"L'hôte virtuel" (virtual host) se réfère à la pratique de maintenir plus d'un serveur sur une seule machine, ceux-ci étant différenciés par leur nom d'hôte apparent. Par exemple, il est souvent désirable, pour des sociétés partageant un serveur web, d'avoir leurs propres espaces avec des serveurs accessibles comme www.company1.com et www.company2.com, et sans demander à l'utilisateur de connaître des informations supplémentaires sur le chemin à suivre.

Apache a été l'un des premiers serveurs à contenir et supporter des hôtes virtuels right out of the box, mais depuis la base HTTP (HyperText Transport Protocol) standard ne permet aucune

méthode pour le serveur consistant à déterminer le nom de l'hôte virtuel à qui le client à adresser une requêtes, le support d'hôte virtuel d'Apache a demandé que l'on attribue à chaque serveur une adresse IP. La documentation utilisant cette nouvelle approche est disponible (celle-ci marche très bien) is available.

Alors que l'approche décrite ci-dessus fonctionne, avec un espace disponible pour les adresses IP diminuant, et le nombre de domaines utilisés par des sociétés augmentant, vous constatez que ce n'est pas la meilleure solution. Il est, en effet, difficile de réaliser ceci sur certaines machines. Le protocole HTTP/1.1 contient une méthode pour le serveur, afin que celui-ci puisse identifier le nom de l'hôte auquel un message a été adressé. La version 1.1 d'Apache et les versions suivantes supportent également cette approche aussi bien que la méthode traditionnelle des adresse IP pour des hôtes virtuels. L'avantage d'utiliser le nouveau support d ' hôte virtuel est que l'on peut avoir un nombre pratiquement illimité de serveurs, faciles d'utilisation, simples à configurer, et ne demandant aucun hardware ou software supplémentaire. Le principal inconvénient est que le navigateur de l'utilisateur doit supporter cette partie du protocole. Les dernières versions de beaucoup de navigateurs (Netscape Navigator 2.0 et les dernières versions incluent) le font à l'exception des plus anciennes. Ceci peut causer des problèmes, alors qu'une solution possible est expliquée ci-dessous.

Named-Based Virtual Hosts

```
NameVirtualHost 192.168.10.2

<VirtualHost www.stephane.gill>
    ServerName www.stephane.gill
    ServerAdmin webmestre@stephane.gill
    DocumentRoot /var/www/site1
    ErrorLog /var/log.httpd_site1/error_log
    TransferLog /var/log.httpd_site1/access_log
</VirtualHost>

<VirtualHost www.charles.gill>
    ServerName www.charles.gill
    ServerAdmin webmestre@charles.gill
    DocumentRoot /var/www/site2
    ErrorLog /var/log.httpd_site2/error_log
    TransferLog /var/log.httpd_site2/access_log
</VirtualHost>
```

NameVirtualHost

La directive NameVirtualHost associe une adresse IP à un numéro de port, si nécessaire, pour tout hôte virtuel portant un nom. La configuration d'hôtes virtuels nommés permet à un Serveur HTTP Apache de servir différents domaines sans devoir pour ce faire utiliser de multiples adresses IP.

Note : L'utilisation de tout hôte virtuel nommé fonctionne seulement avec des connexions HTTP non-sécurisées. Si vous devez employer des hôtes virtuels avec un serveur sécurisé, utilisez plutôt des hôtes virtuels basés sur l'adresse IP.

Afin d'activer des hôtes virtuels basés sur le nom, supprimez le caractère de commentaire de la directive de configuration NameVirtualHost et ajoutez l'adresse IP correcte. Ajoutez ensuite des conteneurs VirtualHost supplémentaires pour chaque hôte virtuel.

VirtualHost

Des balises <VirtualHost> et </VirtualHost> permettent de créer un conteneur soulignant les caractéristiques d'un hôte virtuel. Le conteneur <VirtualHost> accepte la plupart des directives de configuration.

Un ensemble de conteneurs VirtualHost commentés est fourni dans httpd.conf et illustre l'ensemble minimum de directives de configuration nécessaire pour chaque hôte virtuel.

ServerName

Utilisez ServerName pour définir un nom d'hôte et un numéro de port (en accord avec la directive Listen) pour le serveur. La directive ServerName ne doit pas forcément correspondre au nom d'hôte de l'ordinateur. Par exemple, le serveur Web pourrait être www.example.com bien que le nom d'hôte de l'ordinateur soit foo.example.com. La valeur spécifiée dans ServerName doit être un nom de domaine (ou DNS, de l'anglais 'Domain Name Service') valide qui peut être résolu par le système — ne vous contentez surtout pas d'en inventer un.

Ci-dessous figure un exemple de directive ServerName:

```
ServerName www.example.com:80
```

Lors de la détermination d'un ServerName, assurez-vous que son adresse IP et son nom de serveur sont bien inclus dans le fichier /etc/hosts.

ErrorLog

ErrorLog spécifie le fichier dans lequel sont consignées les erreurs du serveur. La valeur par défaut pour cette directive est /var/log/httpd/error_log.

IP-Based Virtual Hosts

```
<VirtualHost 192.168.10.2>
    ServerName www.stephane.gill
    ServerAdmin webmestre@stephane.gill
    DocumentRoot /var/www/site1
    ErrorLog /var/log.httpd_site1/error_log
    TransferLog /var/log.httpd_site1/access_log
</VirtualHost>

<VirtualHost 192.168.10.3>
    ServerName www.charles.gill
    ServerAdmin webmestre@charles.gill
    DocumentRoot /var/www/site2
    ErrorLog /var/log.httpd_site2/error_log
    TransferLog /var/log.httpd_site2/access_log
</VirtualHost>
```

Mixed Name/IP-Based Virtual Hosts

```
NameVirtualHost 192.168.10.2

<VirtualHost www.stephane.gill>
    ServerName www.stephane.gill
    ServerAdmin webmestre@stephane.gill
    DocumentRoot /var/www/site1
    ErrorLog /var/log.httpd_site1/error_log
    TransferLog /var/log.httpd_site1/access_log
</VirtualHost>

<VirtualHost www.charles.gill>
    ServerName www.charles.gill
    ServerAdmin webmestre@charles.gill
    DocumentRoot /var/www/site2
    ErrorLog /var/log.httpd_site2/error_log
    TransferLog /var/log.httpd_site2/access_log
```

```
</VirtualHost>

<VirtualHost 192.168.10.3>
    ServerName www.charles.gill
    ServerAdmin webmestre@charles.gill
    DocumentRoot /var/www/site2
    ErrorLog /var/log.httpd_site2/error_log
    TransferLog /var/log.httpd_site2/access_log
</VirtualHost>
```

Port-Based Virtual Hosting

```
Listen 80
Listen 8080

<VirtualHost 192.168.10.2:80>
    ServerName www.stephane.gill
    ServerAdmin webmestre@stephane.gill
    DocumentRoot /var/www/site1
    ErrorLog /var/log.httpd_site1/error_log
    TransferLog /var/log.httpd_site1/access_log
</VirtualHost>

<VirtualHost 192.168.10.2:8080>
    ServerName www.charles.gill
    ServerAdmin webmestre@charles.gill
    DocumentRoot /var/www/site2
    ErrorLog /var/log.httpd_site2/error_log
    TransferLog /var/log.httpd_site2/access_log
</VirtualHost>
```

Authentification

L'authentification est un principe simple. Elle permet de limiter l'accès à un site Web à des utilisateurs autorisés. Pour ce faire, il faut d'abord créer une base de données des utilisateurs avec un mot de passe pour chacun d'eux. Puis ensuite, modifier le fichier de configuration de Apache de façon à activer l'authentification sur les répertoires à protéger.

Création des mots de passe

La base de données contenant les mots de passe des utilisateurs est stockée dans un fichier dont le format est semblable à celui du fichier */etc/passwd* de Linux. Ce fichier est géré à l'aide de la commande *htpasswd* et doit être créer à l'extérieur du répertoire pointé par *DocumentRoot*. L'exemple suivant présente la création du fichier à l'aide de la commande *htpasswd* :

```
# htpasswd -c /var/www/MotDePasse DeBoss
New password:
Re-type new password:
Adding password for user DeBoss
```

L'ajout d'utilisateurs dans le fichier de mot de passe est effectué comme dans l'exemple suivant :

```
# htpasswd  /var/www/MotDePasse PetitBoss
New password:
Re-type new password:
Adding password for user PetitBoss
```

Le fichier est lisible et modifiable avec un éditeur de texte mais les mots de passe sont cryptés.

```
# more /var/www/MotDePasse
DeBoss:Bz9Wp3o8tchOM
PetitBoss:.arflSl7hToLE
```

Directive d'authentification

La configuration du serveur consiste à définir un espace dans lequel l'authentification est nécessaire. Ceci est effectué dans un bloc *<Directory>* en spécifiant les directives suivantes :

```
<Directory /var/www/html/monrep>
    AuthType Basic
    AuthName nom
    AuthUserFile /var/www/MotDePasse
    AuthGroupFile /var/www/Groupe
    require valid-user
<Directory>
```

AuthType

Cette directive sélectionne le type d'authentification pour un répertoire. Seul le type *Basic* est actuellement implémenté. (*Digest* est maintenant disponible, codé MD5).

AuthName

Cette directive indique le nom du schéma d'autorisation pour un répertoire. Ce schéma sera donné au client de sorte que l'utilisateur sache quel nom et quel mot de passe envoyer.

AuthUserFile

La directive *AuthUserFile* définit le nom du fichier texte dans lequel est enregistrée une liste de noms d'utilisateurs et des mots de passe qui leur sont associés dans le but d'une authentification d'accès. *nomFichier* est le chemin d'accès absolu au fichier d'utilisateurs.

Chaque ligne de ce fichier spécifie un nom d'utilisateur suivi d'un deux-points, puis du mot de passe sous sa forme cryptée par la fonction crypt(). Le comportement en cas de multiples occurrences du même nom d'utilisateur reste indéterminé. Notez que dès que le fichier utilisateurs est de grande taille, cette méthode se révèle très peu performante ; on préfèrera utiliser *AuthDBMUserFile* à la place.

AuthGroupFile

La directive *AuthGroupFile* définit le nom du fichier texte contenant la liste des groupes d'utilisateurs destinée à servir de base à l'authentification d'accès. *nomFichier* est le chemin d'accès absolu du fichier de groupes.

Chaque ligne du fichier de groupes d'utilisateurs contient le nom d'un groupe suivi par un deux-points, puis par la liste des noms d'utilisateurs assignés à ce groupe, lesquels sont séparés par des espaces. Exemple :

```
mongroupe: bob jacques anne
```

Notez que dès que le fichier de groupes est de grande taille, cette méthode se révèle très peu performante ; on préfèrera utiliser *AuthDBMGroupFile* à la place.

require

Cette directive choisie quels utilisateurs autorisés peuvent accéder à un répertoire. Les syntaxes valides sont :


```
require user utilisateur utilisateur ...
```

Seuls les utilisateurs nommés peuvent accéder au répertoire.

```
require group nomGroupe nomGroupe ...
```

Seuls les utilisateurs des groupes cités peuvent accéder au répertoire.

```
require valid-user
```

Tout utilisateur reconnu peut accéder au répertoire (par opposition aux non utilisateurs).

Si *require* apparaît dans une section <*Limit*>, alors les restrictions ne sont appliquées qu'aux méthodes http mentionnées. Autrement, toutes les méthodes http sont restreintes. Exemple :

```
AuthType Basic
AuthName unDomaine
AuthUserFile /web/users
AuthGroupFile /web/groups
<Limit GET POST>
require group admin
</Limit>
```

Fichier .htaccess

Les fichiers .htaccess sont, en fait, des fichiers de configuration d'Apache, permettant de définir des règles dans un répertoire et dans tous ses sous répertoires (qui n'ont pas de tel fichier à l'intérieur). On peut les utiliser pour protéger un répertoire par mot de passe, ou pour changer le nom ou l'extension de la page index, ou encore pour interdire l'accès au répertoire.

Exemple de fichier *.htaccess* pour protéger l'accès à un répertoire par un mot de passe :

```
# more .htaccess
AuthType Basic
AuthName darkness
AuthUserFile /var/www/MotDePasse
require valid-user
```

Dans le fichier */etc/httpd/conf/httpd.conf* c'est la directive *AccessFileName* qui permet de définir lele nom du fichier qui sera reconnu comme étant un fichier de restriction d'accès. Par défaut, c'est .htaccess :

```
AccessFileName .htaccess
```

De plus, il est parfois nécessaire de modifier la directive *AllowOverride* de façon à autoriser l'utilisation des fichiers *.htaccess*. Voici un exemple :

```
AllowOverride AuthConfig
```

ou

```
AllowOverride All
```