TP Serveur Web : appache 2.0.40

October 8, 2007

Attention :

- Lors de l'installation d'Apache, les packages suivants son instalés:

```
* apache2-common  : fichiers d'installation pour Apache et apache-mod_perl
* apache2-modules : ensemble des modules pour Apache
* apache-conf     : fichiers de configuration de Apache
* apache2         : le dmon Apache
* apache-manuel (pas obligatoire mais peut se r\'ev\'eler utile).
```

- Les fichiers de configuration se trouvent généralement dans `/etc/httpd/conf/httpd.conf` et `/etc/httpd/conf.d/`.
- **Avant toute modification du fichier** `httpd.conf`**, faites une copie de sauvegarde vers** `httpd.conf.old`**, avec** `cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.old`.
- Après toute modification de la configuration du serveur il faut le relancer par `service httpd restart`.
- Les navigateurs mémorisent dans leur cache les pages précédemment chargées.
- Penser à faire des captures de trames avec Wirechark.
- **Une documentation e ligne est disponible sur** `http:/localhost/manual`.

1 Installation/test de base du serveur

a) En vérifiant les valeurs des paramètres des étapes décrites dans l'annexe A, lancez le serveur Web sur votre machine. Pour tester son fonctionnement vous utiliserez votre navigateur préféré à l'URL `http://localhost/`. Visualisez rapidement les messages d'erreurs et les messages d'informations dans `/var/log/httpd/error_log` et `access_log`.

b) A l'aide de la commande **ps auxwf | grep httpd**, vérifier qu'il y a bien plusieurs daemons httpd. Combien ? Quels en sont les propriétaires ? À quel paramètre correspond t-il ?

c) Dans quel répertoire sont installés par défaut les pages HTML ?

2 Configuration avancé du serveur

Éditer le fichier `/etc/httpd/conf/httpd.conf`

a) Modifier le port écouté par le serveur (par exemple 8080). Donner la méthode de configuration et l'URL pour le port 8080. Tester avec votre navigateur.

[b)] Modifier également l'interface d'écoute en créant une alias `eth0:1`. Tester avec votre navigateur.

c) Vérifier que le port 80 n'est plus écouté (avec `nmap -F 'adresse-IP' 'nom-machine'`). Remettre ensuite le port 80.

d) On veut que l'URL `http://localhost/` corresponde automatiquement à la page d'index (index.html). Que faut-il modifier ? Le tester.

2.1 Premier script PHP

Vous allez réaliser votre premier test en PHP.

a) Commencez par Installer PHP et ensuite créez un sous répertoire `/var/www/html/test/`.

b) Créez dans ce sous répertoire un fichier `test.php` contenant :

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

c) Sauvegardez et chargez le fichier dans un navigateur (URL `http://localhost/test/test.php`).

- tapez simplement `http://localhost/test/` dans la barre d'URL de votre navigateur, celui-ci affiche la liste des fichiers créés.
- Renommez le fichier vers `index.php`. Que constatez-vous ? Le contenu du sous rpertoire est-il toujours visible ?
- effacez le ficher index.php, l'affichage de la liste des fichiers est à nouveau possible. C'est pratique pour éviter qu'un visteur de passage puisse consulter le contenu d'un sous-répertoire.

2.2 Premier script CGI

a) Dans le répertoire `/var/wwww/cgi-bin`, créer et rendre exécutable le script `test-cgi` ci-dessous. Penser à donner les droits d'écriture et d'éxecution sur ce répertoire et sur le fichier.

```
#!/bin/sh
echo "Content-type: text/plain"
echo
echo " <H1>R\'esultat de l'excution du script</H1>"
```

b) Charger l'URL `/cgi-bin/test-cgi`. Résultat ? Que faut-il modifier au script pour générer une page HTML ?

c) Modifier ce script pour qu'il affiche les informations suivantes:

- coté **client** : adresse IP, port TCP et nom du navigateur.
- coté **serveur** : nom de la machine, nom de la machine servant le WEB, numro de port TCP.
- **Remarque**: *les variables d'environnement correspondantes peuvent être récupérées grâce au script PHP précédent.*

3 Configuration des répertoires personnels

Vous allez mettre en place un accès pour les utilisateurs du système. Ceux-ci auront la possibilité de mettre leurs pages personnelles dans leurs répertoires privés (i.e.,`public_html`).

a) Relevez dans le fichier de configuration le nom du répertoire dans lequel doivent être stockées les pages personnelles. Quelle(s) directive(s) il faut utiliser ?

b) Créez un compte utilisateur, par exemple **user1**. Créez le répertoire du site web personnel dans `/home/user1/`. Installez une page HTML, `index.html`, dans le répertoire.

c) Testez l'accessibilité de la page en utilisant l'URL `http://localhost/~user1`.

d) Pour pouvoir accéder aux fichiers de `public_html`, le répertoire `/home/user1/public_html` doit avoir les droits de traverse (+x). Le fichier `index.html` doit lui avoir les droits de lecture.

4 Les hôtes virtuels

Un hôte virtuel est un site web hébergé sur votre serveur et qui est identifié comme un alias. Il se comporte exactement comme un second serveur s'exécutant sur la même machine, mais apparaissant comme le seul serveur disponible pour un client qui se connecte sur cet hôte virtuel.
Le serveur Apache permet de configurer différents types d'hébergement virtuels :

- **Les hôtes virtuels par adresse IP** sont identifiés par l'adresse IP à laquelle arrivent les requêtes des clients. Chaque hôte virtuel de ce type dispose de sa propore adresse IP.
- **Les hôtes virtuels par nom** tirent parti de l'en-tête `Host` (requêtes HTTP/1.1) pour identifier un hôte virtuel par le nom du serveur auquel s'adresse le client.

La directive `<VirtualHost>` contient les directives qui s'appliquent à un hôte virtuel donné, qu'il soit par adresse ou par nom. Ils sont gérés par le module `mod_virtual`.
Penser à consulter l'aide en ligne sur `http:/localhost/manual`, pour la configuration des hôte virtuels.

4.1 Configuration d'un hôte virtuel par adresse IP

Pour mettre en place les hôtes virtuels par adresse, on utilise le conteneur `<VirtualHost adresse_IP>` pour regrouper les directives qui ne s'appliquent qu'à l'hôte virtuel défini par `adresse_IP`.
Pour créer deux hôtes virtuels par adresse sur le serveur Apache, sur la même interface réseau, on procède comme suit :

1. On ajoute deux interfaces virtuelles à la carte réseau du serveur (`eth0`) :

   ```
   #/sbin/ifconfig eth0:0 192.168.1.1
   #/sbin/ifconfig eth0:1 192.168.1.2
   ```

2. On définit un hôte virtuel par adresse pour chaque interface virtuelle créée sur le serveur :

   ```
   <VirtualHost 192.168.1.1>
   ServerName vhost1.iut-caen.fr
   DocumentRoot /var/www/html/vhost1
   </VirtualHost>

   <VirtualHost 192.168.1.2>
   ServerName vhost2.iut-caen.fr
   DocumentRoot /var/www/html/vhost2
   </VirtualHost>
   ```

3. On se connecte au premier site en utilisant l'URL suivante : `http://192.168.1.1/`. Le serveur offre les documents stockés dans `/var/www/html/vhost1`.
4. Pour se connecter au même site avec l'URL `http://vhost1.iut-caen.fr/`, il faut créer un enregistrement dans le serveur DNS du domaine, qui fait correspondre l'adresse IP à un nom d'hôte. Il est également possible d'utiliser le fichier `hosts` pour faire la résolution.
5. Changez à présent le numéro de port du second hote virtuel (192.168.1.2:8080).

4.2 Configuration d'un hôte virtuel par nom

On souhaite configurer un serveur web pour `namedvh1.iut-caen.fr` et `namedvh2.iut-caen.fr`, sur une machine ayant une seule adresse IP. Voici les étapes à suivre pour configurer les deux hôtes virtuels par nom.

1. On utilise la directive spéciale `NameVirtualHost` pour indiquer l'adresse IP sur lequel le serveur recevra toutes les requêtes en provenance des hôtes virtuels par nom.

```
NameVirtualHost 192.168.1.1
```

2. On définit les hôtes virtuels par nom associés à cette adresse.

```
<VirtualHost 192.168.1.1>
ServerName namedvh1.iut-caen.fr
DocumentRoot /var/www/html/NamedVH1
</VirtualHost>

<VirtualHost 192.168.1.1>
ServerName namedvh2.iut-caen.fr
DocumentRoot /var/wwww/html/NamedVH2
</VirtualHost>
```

   Dès qu'Apache lit la directive `NameVirtualHost` dans `httpd.conf`, il met en place une table d'hôtes virtuels pour l'adresse indiquée et y ajoute les noms indiqués par la directive `ServerName` au fur et à mesure qu'il traite les configurations d'hôtes virtuels qui suivent.

3. Il faut ensuite entrer les noms des ces machines (les hôtes virtuels par nom) comme des enregistrements de noms canoniques, ou CNAME, dans le serveur DNS du domaine.

4. On se connecte à un des sites en utilisant le nom indiqué par la directive `ServerName`. Par exemple : `http://namedvh1.iut-caen.fr`.

5. Changez à présent le numéro de port du second hote virtuel (192.168.1.2:8080).

5 Droits d'accès et authentification

5.1 Authentification de base

HTTP permet de gérer les droits d'accès, en utilisant une méthode d'authentification permettant de sécuriser l'accès à un répertoire ou à des fichiers après avoir demandé à l'utilisateur un nom et un mot de passe.
C'est une méthode qui n'est pas sécurisée car elle transmet les mots de passe en clair sur le réseau.
Les directives suivantes permettent de mettre en œuvre ce type d'authentification :

- `AuthName`, définit ce qui sera affiché au client pour lui demander son nom et son mot de passe,
- `AuthType`, définit le type d'authentification utilisée pour le mot de passe. Si `Basic` est utilisé, le mot de passe est en texte clair, si `Digest` est spécifié, le mot de passe est crypté par MD5 avant d'être envoyé sur le réseau (penser á faire des captures de trames),
- `AuthUserFile`, définit le chemin d'accès vers le fichier qui contient la liste des utilisateurs et des mots de passe,
- `AuthGroupFile`, définit le chemin d'accès vers le fichier qui contient la liste des groupes,
- `require`, permet de définir quelles personnes ou groupes qui ont une permission d'accès. Le mot clé `valid-user` précise que l'on autorise uniquement les personnes identifiés. La directive `require user {username}` permet de préciser explicitement le nom de personnes autorisées à s'identifier.

Apache fournit un outil permettant de génerer des mots de passe cryptés, il s'agit de l'utilitaire `htpasswd`.

a) Consultez la syntaxe de `htpasswd`, puis créez un nouveau fichier de mots de passe pour l'utilisateur *jsmith* (avec comme mot passe *smith*).

   Le fichier devra être stocké dans `/var/www/html/secure`.

b) Avec le conteneur `Directory` définissez un accès sécurisé (uniquement les personnes identifiées) au répertoire `/var/wwww/html/usage`. Pensez à relancer le serveur.

c) Testez la configuration en utilisant l'URL `http://localhost/usage`. Corrigez les erreurs que vous pouvez rencontrer (permission d'accès ...)

5.2 Méthode d'authentification sécurisée, HTTP sur SSL

Cette partie est facultative. Elle sera traitée en détails dans le module complémentaire TR-C8. Sécuriser la communication entre un serveur Web et un navigateur client est parfois utile, en particulier lorsqu'il s'agit d'informations sensibles (commerce élecronique, etc.). Le serveur Apache emploie SSL (Secure Socket Layer) pour sécuriser les transactions HTTP. `HTTPS` est le terme utilisé pour désigner un protocole HTTP "sur" SSL.

SSL utilise une technique de chiffrement à clef publique. Pour sécuriser la connexion, le serveur envoie au client une clef publique que celui-ci utilisera pour chiffrer les données que seul le serveur pourra déchiffrer grâce à sa clef privée. L'étape suivante consiste éventuellement pour le client à envoyer de manière chiffrée sa clef publique. OpenSSL fournit les outils permettant de créer les clés privé et publiques nécessaires.

Sous Mandrake, on utilise un script (`/usr/lib/ssl/apache2-mod_ssl/gentestcrt.sh`) servant à la génération des fichiers `server.key` et `server.cert`, correspondant respectivement à la clé privée du serveur et au certificat autosigné par le serveur. Ils se trouvent (en général) dans `/etc/ssl/apache2/`.

a) Quel module dans Apache gère cela ? Relevez le port par défaut utilisé.

b) Sous Fedora, retrouvez le script permettant de générer ces clés.

c) Supprimez les fichiers `server.key` et `server.cert`. Créez à nouveau ces deux fichiers, les copiez dans le répertoire de configuration de SSL. Testez le serveur SSL en utilisant l'URL `https://localhost`.

d) Expliquez les différentes étapes utilisées lors de la génération du certificat.

A Configuration du serveur HTTP

Le fichier `httpd.conf`, comporte toutes les directives de configuration. C'est le fichier principal de configuration du serveur. On trouve par exemple, le numéro de port utilisé pour les connexions, et d'autres informations à mettre à jour pour configurer le serveur. Pour installer un serveur supportant PHP un fichier de configuration dédié est fournit : `mod_php.conf`, de même pour installer un serveur https (`mod_ssl.conf`).

- `User` définit l'utilisateur Linux sous le compte duquel s'exécuteront les processus fils chargés de répondre aux requêtes des clients.

  Le comportement par défaut d'Apache consiste à donner tous les processus fils à l'utilisateur `nobody`. Vous pouvez également créer un compte utilisateur réservé au serveur web.

- `Group` sert à changer le groupe propriétaire des processus fils chargs de répondre aux requêtes des clients. Un bon choix est de donner au serveur droits de groupe `nogroup`.

- `ServerAdmin` est l'e-mail du webmaster.

- `ServerRoot` précise le chemin vers le répertoire de configuration (conf) et vers le répertoire de contrle (logs).

- `ServerName` est le nom d'hôte de la machine sur laquelle tourne le serveur. Il faut que cela soit un nom reconnu du DNS. Dans le cas d'un système n'hébergeant qu'un seul site web, **cette directive vaut généralement le nom d'hôte et de domaine de ce serveur**.

- `StartServers`. Au démarrage il se crée `StartServers`.

- `MaxClients` est le nombre maximal de clients.

- `MaxRequestsPerChild` est le nombre de requêtes qu'un serveur peut traiter avant de mourir. Ceci permet de toujours avoir des serveurs récents.

B Configuration des services et de l'arborescence

B.1 Chemins vers les documents

- `DocumentRoot` spécifie le répertoire racine de l'arborescence des fichiers distribués par apache. Il contient les fichiers qu'Apache fournit lorsqu'il reçoit des requêtes avec l'URL /.

  Il est possible d'utiliser la valeur que fournit Apache, mais en général on utilise le répertoire `/home/httpd/html`.

  Si vous changez la valeur de cette directive, vous devez également modifier la valeur qui apparaît dans la directive `<Directory>`.

- `UserDir` spécifie le nom du répertoire où chaque utilisateur peut mettre ses pages personnelles. Par exemple, l'utilisateur *pujasp* met ses pages dans ∼pujasp/public_html/. On accede aux pages par l'URL http:// monserveur/∼pujasp/

- `Alias` permet de lier une URL à un chemin.

  Par exemple `Alias /html/ /home/httpd/html/` spécifie que l'URL `http://monserveur/html/` correspond au répertoire `/home/httpd/html/`.

- `ScriptAlias` indique un répertoire contenant des scripts exécutables, par exemple des programmes CGI pouvant être appelés à partir d'un navigateur web.

  Par exemple, `ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/`.

- `AccessFileName` spécifie le nom du fichier de gestion de l'accès aux répertoires.

- `Redirect` permet de rediriger les requêtes vers un autre serveur (pages qui changent de site).

- `ErrorDocument` sont des directives qui permettent de personnaliser l'action associée aux messages d'erreur du serveur.

B.2 Indexation

- `DirectoryIndex` précise le comportement d'Apache lorsqu'il reçoit une requête correspondant à un répertoire sur le serveur.

  Cette directive active l'utilisation d'une indexation sur un répertoire. Elle permet de préciser les fichiers d'index. Le serveur parcours cette liste et renvoie le premier fichier trouvé. S'il n'y en a pas il génère automatiquement un index.

- `IndexOptions` permet de positionner les options pour créer un index ou un listing de répertoire amélioré :

  1. `FancyIndexing`: indexation plus agréable ;
  2. `IconsAreLinks`: Rend les icône des fichiers cliquables ;
  3. `SuppressLastModified`, `SuppressSize`, `SuppressDescription`: supprime l'affichage de la date de dernière modification, de la taille ou de la description des fichiers.

- `AddIcon`... indique l'icône qui est associé à un fichier lorsqu'on utilise un index amélioré pour afficher le contenu d'un répertoire.

- `IndexIgnore` interdit que certains fichiers apparaissent dans l'index automatique.

C Configuration de l'accès aux documents

La configuration d'un document peut se situer à plusieurs niveaux de hiérarchies :

- <Directory repertoire> ... </Directory> délimitent l'action des primitives de configuration à un répertoire et ses sous répertoire.
- <Files fic1 fic2> ... </Files> à certains fichiers.
- <Location url> ... </Location> à certaines URL.

Voici un exemple autorisant l'accès, pour une machine donnée, à l'URL `/Document/Privé` :

```
<Location /Document/Priv>
 order deny,allow
 deny from all
 allow from nom_machine
</Location>
```

`Options` est une directive qui contrôle la disponibilité de certaines fonctionnalités du serveur dans un répertoire particulier. Cette directive peut valoir `None`, ou l'une des valeurs suivantes :

- `Indexes` autorise la génération automatiquement d'index si l'URL demandé est un répertoire et s'il n'y a pas de fichier `DirectoryIndex` (`index.htlm` par exemple) ;
- `ExecCGI` autorise le lancement des scripts ;
- `FollowSymLinks` autorise le serveur à suivre les liens symboliques ;
- `SymLinksIfOwnerMatch` autorise le suivit des liens si le propriétaire ne change pas ;

`AllowOverride` Précise les éléments de la configuration que le fichier d'accès (.htaccess) peut modifier :

- `AuthConfig` toute configuration d'autorisation nominative (contrôle par mot de passe) ;
- `FileInfo` toute option de contrôle des documents (langue, type, erreur,) ;
- `Indexes` toute option sur les index, en-tte, bas de page, etc. ;
- `Limit` précise quelles machines peuvent accéder aux documents ;

`order` précise l'algorithme de vérification des accès :

- `allow,deny` autorise puis interdit
- `deny,allow` interdit puis autorise

`allow from`

- `all` toutes les machines peuvent accéder à cette partie du site
- `10.40.103 10.40.103` seules des machines sur les sous-réseaux 10.40.103.0 et 10.40.104.0 peuvent accéder ;
- `fr edu com` seules des machines des domaines .fr .edu ou .com peuvent accéder ;
- `gtr.iut-caen..fr` seules des machines du domaine gtr.iut-caen.fr peuvent accéder.

`deny from machine` ou `rśeau` ou `all` : Idem.

Running several name-based web sites on a single IP address.

Your server has a single IP address, and multiple aliases (CNAMES)
point to this machine in DNS. You want to run a web server for
www.example1.com and www.example2.org on this machine.

Creating virtual host configurations on your Apache server does not
magically cause DNS entries to be created for those host names. You
must have the names in DNS, resolving to your IP address, or nobody
else will be able to see your web site. You can put entries in your
hosts file for local testing, but that will work only from the machine
with those hosts entries.

Server configuration

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
DocumentRoot /www/example1
ServerName www.example1.com

# Other directives here

</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/example2
ServerName www.example2.org

# Other directives here

</VirtualHost>
```

The asterisks match all addresses, so the main server serves no requests.
Due to the fact that www.example1.com is first in the configuration file,
it has the highest priority and can be seen as the default or primary server.
That means that if a request is received that does not match one of the specified
ServerName directives, it will be served by this first VirtualHost.
Note

You can, if you wish, replace * with the actual IP address of the system.
In that case, the argument to VirtualHost must match the argument to NameVirtualHost:

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
# etc ...
```

However, it is additionally useful to use * on systems where the IP address
is not predictable - for example if you have a dynamic IP address with your ISP,
and you are using some variety of dynamic DNS solution. Since * matches any IP address,
this configuration would work without changes whenever your IP address changes.

The above configuration is what you will want to use in almost all name-based virtual hosting situations. The only thing that this configuration will not work for, in fact, is when you are serving different content based on differing IP addresses or ports.