- COMMENT LIRE CE DOCUMENT ?

Ce document est volumineux et risque de vous décourager, mais … il est quand même fait pour vous faire gagner du temps pour vos TP : voici donc la façon la plus optimale de le lire :

- **Chapitre 1** : (conseillé) vous explique le projet global et le 1.3 le matériel à utiliser
- **Chapitre 2** : (facultatif) vous rappelle quelques généralités d'Apache et Linux que vous connaissez ?
- **Chapitre 3 et 4** : (impératif) AVANT de commencer les TP. Si vous ne faites pas cet effort là … vous allez perdre du temps et risquer de faire des manipulations approximatives
- **Chapitre 5** : (au besoin) Référez-vous à ces pages lorsque vous recherchez des explications sur une directive (comme quand vous cherchez dans un dictionnaire …). Vous pouvez aussi consulter le site d'Apache2 qui vous fournira des informations beaucoup plus précises.
- **Chapitre 6 et 7** : (plus tard) Ces deux sections sont faites pour vous aider à réaliser les TP. Vous les lirez lorsque vous en aurez besoin.
- **Chapitre 8 :** c'est le sujet du TP … c'est vous qui voyez ! ☺

Apache2 : un serveur HTTP

1 Introduction

L'objectif du TP

Ce TP vous permettra d'effectuer les manipulations de bases vous permettant de configurer un serveur HTTP sur une machine. Vous aborderez également les notions de sécURLté élémentaires proposées par ce serveur.

1.1 Généralités et Rappels de cours

Le rôle principal d'un serveur web est très simple : répondre à des requêtes client de la forme : « je voudrais consulter le fichier X se trouvant dans le dossier Y du disque dur de votre machine ». Le serveur renvoie alors ce fichier en utilisant un protocole de niveau applicatif appelé HTTP (Hyper Text Transfert Protocol), sur le protocole de transport TCP/IP.

Au fur et à mesure des améliorations des spécifications d'un serveur http, il a été prévu de répondre à des requêtes plus complexes, comme par exemple : « je voudrais consulter le fichier X, correspondant au résultat de l'exécution du programme Y se trouvant sur le disque dur de votre machine ». Ces nouvelles fonctionnalités permettent ainsi d'interroger des bases de données, comme sur tous les bons moteurs de recherche.

Dans le premier exemple, on parlera de page statique (le fichier à renvoyer existe déjà sur le disque dur du serveur). Dans le deuxième cas, on parlera de page dynamique (le fichier renvoyé est construit par un programme « à la demande » du client).

Pour information, sachez que le format HTML est juste un langage de mise en forme, que le client est capable d'interpréter. Un serveur quant à lui ne doit pas se soucier de renvoyer un fichier au format HTML, en texte brut, une image ou un son … cela reste le problème du client de savoir comment il va le prendre en compte.

1.2 Mise en place de l'environnement de TP

Pour effectuer les manipulations de ce TP, il vous faut deux machines : vous utiliserez un PC Linux pour installer le **serveur** HTTP, et n'importe quel **client** HTTP (« Firefox », Chrome, Edge, ...) pour effectuer les tests. Vous pouvez, si cela vous arrange, utiliser la même machine pour les deux rôles (en vous assurant de ne pas consulter les fichiers html localement sur le disque dur, mais de bien interroger le serveur Apache …)

2 Généralités sur le serveur http APACHE

2.1 Présentation

Le serveur « Apache » a longtemps été le plus performant et le plus utilisé des serveurs HTTP. Cette notoriété est essentiellement due à sa gratuité totale, sa maintenance efficace et permanente, son support technique en ligne (www.apache.org), ses performances, sa modularité (il est très facile de programmer de nouvelles fonctionnalités et de les intégrer en tant que module), sa portabilité (il existe des versions pour tous les Unix, pour Windows, ...) et surtout le fait qu'il soit « open source » c'est à dire que tout le monde soit en mesure de consulter analyser et modifier les lignes de programme qui le constituent.

Néanmoins, il est concurrencé depuis quelques années par NGINX qui présente les mêmes avantages et a été essentiellement conçu pour mieux gérer les très fortes charges (nb de connexions en simultané). NGINX supplante largement APACHE sur ce point, mais présente encore quelques manques de fonctionnalités par rapport à son concurrent historique. Etant donné que les sites web de la taille de Google ou Amazon ne représentent pas la majorité du web, nous avons choisi de continuer à étudier Apache en TP, sachant qu'une fois les concepts de base acquis, le passage de l'un à l'autre ne devraient pas vous poser de problèmes particuliers.

2.2 Lancement, arrêt et configuration

Comme tout service, le serveur apache se démarre, s'arrête, se redémarre ou donne son état grâce à la commande :

```
service apache2 [start | stop | restart | status]
```

Cette commande est devenue obsolète et disparaîtra bientôt, au profit de la nouvelle commande :

```
systemctl [start | stop | restart | status] apache2
```

Mieux vaut donc vous y habituer dès à présent.

En démarrant, Apache puise ses informations de configuration dans : `/etc/apache2/apache2.conf`.

3 Les fichiers de configuration

3.1 Généralités

Cette section est d'une importance **CAPITALE**. La structure des fichiers de configuration d'Apache, n'est pas compliquée, mais il est nécessaire de la comprendre parfaitement pour ne pas faire d'erreurs qui pourraient vite avoir des conséquences importantes (voire dramatiques) sur le comportement du serveur.

Il vous est donc largement conseillé d'en prendre connaissance avec attention et de vous assurer de tout avoir bien compris. Cela vous permettra de réaliser beaucoup plus facilement tous les TP.

Le SEUL fichier de configuration pris en compte par apache est le fichier « /etc/apache2/apache2.conf », mais ce fichier comporte des directives « Include » ou « IncludeOptional » qui permettent d'inclure d'autres fichiers. En conséquence, la configuration d'Apache est répartie dans une multitude de fichiers, dont il est impératif de respecter le rôle de chacun, non pas pour que cela fonctionne, (vu qu'au final tout se retrouvera dans un seul fichier …), mais pour vous y retrouver plus tard, lorsque vous souhaiterez modifier la configuration ou surtout si vous travaillez en équipe.

Si on supprime toutes les lignes de commentaire du fichier apache2.conf (on peut faire cela par exemple avec la commande suivante : « cat /etc/apache2/apache2.conf | grep -v ^# » (dans le « grep », ^# retient toute ligne qui commence par # et -v signifie d'inverser la sélection. Ainsi, on conserve toutes les lignes qui ne commence PAS par #), on s'aperçoit qu'il ne reste plus grand-chose !

Les lignes concernant les « Include » sont :

```
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
Include ports.conf
IncludeOptional conf-enabled/*.conf
IncludeOptional sites-enabled/*.conf
```

Mettons fin au suspense tout de suite, la différence entre «Include» et «IncludeOptional» est minime : la deuxième directive ne génère pas de message d'erreur, si le fichier ciblé n'existe pas. Ceci permet d'inclure des fichiers de configuration qui n'existent pas (encore) sans faire hurler Apache au démarrage. Le fichier « ports.conf » étant obligé d'exister, il n'est pris aucune précaution à son égard. En revanche les dossiers « mods-enabled », « conf-enabled » et « sites-enabled » pouvant être vides, il est préférable de se méfier. Toutefois, TOUS les fichiers présents dans ces dossiers seront inclus dans « apache2.conf ». Il est donc possible d'en créer … une infinité !

L'ordre d'inclusion est TRES IMPORTANT, certaines directives pouvant en contredire d'autres. Il correspond à l'ordre des directives d'inclusion dans le fichier apache2.conf, donc telles qu'elles apparaissent ci-dessus. Ainsi les fichiers « .load » du dossier « mods-enabled » seront les premiers et ainsi de suite jusqu'aux fichiers « .conf » du dossier « sites-enabled ».

A l'intérieur d'un même dossier, l'ordre d'inclusion suivra l'ordre alphabétique des noms des fichiers.

Bien entendu, toutes ces inclusions s'intercalent logiquement avec les directives de configurations apparaissant à l'intérieur du fichier « apache2.conf » lui-même.

Vous pouvez donc ajouter, enlever ou changer la configuration à tous les niveaux, vos modifications seront inévitablement répercutées dans le fichier principal de configuration : « apache2.conf » et prises en compte lors du démarrage d'Apache.

Dernier point TRES IMPORTANT : toute modification de la configuration dans l'un quelconque de ces fichiers ET SEULEMENT DANS L'UN QUELCONQUE DE CES FICHIERS ne sera prise en compte qu'APRES le redémarrage d'Apache2.

Autrement dit, toute modification que vous pourrez faire sur le contenu du site web en dehors des fichiers de conf (pages HTML, programmes CGI, …), seront pris en compte sans redémarrage du serveur.

3.2 Les dossiers « -available » et « -enabled »

Le principe utilisé par Apache pour l'activation de certaines ressources est très simple: un dossier porte un nom suffixé par « available » pour indiquer les ressources **disponibles**, et un dossier du même nom suffixé par « enabled » contient une copie des ressources **activées** (c'est-à-dire qui seront inclues dans le fichier apache2.conf). Actuellement trois dossiers fonctionnant sur ce principe existent: « mods » (contenant les modules d'Apache), « sites » (contenant la définition des divers sites hébergés sur le serveur) et « conf » (contenant quelques modèles classiques de configuration).

Par exemple, « mods-available » contient tous les modules fournis avec Apache (vous pouvez en rajouter à loisir si vous savez programmer ...), mais seuls ceux qui sont recopiés sous le dossier « mods-enabled » seront chargés au lancement d'Apache.

Cela dit, au lieu de les recopier (et de perdre ainsi de l'espace inutile sur le disque dur), il suffit de créer dans ce dossier « enabled » un lien symbolique (raccourci) sur le fichier correspondant du dossier « available » pour qu'il soit pris en compte.

MAGIQUE : Les commandes suivantes créent ou suppriment automatiquement ces raccourcis:

a2enmod <module> ou a2dismod <module>

Apache **2** active (**en**able) ou désactive (**dis**able) le **mod**ule précisé

a2ensite <site> ou a2dissite <site>

Apache **2** active (**en**able) ou désactive (**dis**able) le **site** web précisé

a2enconf <conf> ou a2disconf <conf>

Apache **2** active (**en**able) ou désactive (**dis**able) la **conf** précisée

REMARQUE : le dossier sites-enabled contient par défaut un lien 000-default.conf vers le même fichier du dossier sites-available. Il s'agit du site web mis en place automatiquement lors de l'installation d'Apache2.

4 Syntaxe des fichiers de configuration

4.1 Sections et directives

En utilisant un éditeur de texte capable d'afficher plusieurs couleurs un fichier de configuration vous paraîtra beaucoup plus clair ! En mode texte, `vim` est « parfait », mais en mode graphique, utilisez de préférence l'éditeur `gedit`. Vous ferez apparaître les différentes couleurs dans le texte avec les options « Affichage/Mode de Coloration/Balisage/HTML » (gedit n'est pas disponible sur les containers …).

Un fichier de configuration est composé :

- de **DIRECTIVES** (une directive est de la forme <variable> <valeur(s)>). Exemple : **`Port 80`**
  ou encore : **`DirectoryIndex index.html index.htm index.php`**
- de **SECTIONS** (une section commence par <MOTIF> et finit par </MOTIF>). Une section contient elle-même des DIRECTIVES.

A savoir que certaines directives (globales) ne peuvent apparaître qu'en dehors de toute section. D'autres directives ne sont cohérentes qu'à l'intérieur d'une section. Enfin certaines directives peuvent indifféremment apparaitre dans ou hors d'une section. Le fonctionnement des sections sera expliqué plus loin.

Il existe des centaines de directives et de types de sections … pour tout savoir, la bible se trouve ici : https://httpd.apache.org/ et plus particulièrement ici : https://httpd.apache.org/docs/2.4/fr/mod/core.html

4.2 Exemples de directives

Mieux que de longs discours, voyons tout de suite quelques exemple de directives essentielles :

`ServerName www.toto.fr`

La directive ServerName peut être globale ou apparaître dans une section. Elle sert à préciser le nom symbolique du site web hébergé. Attention : il faut bien entendu que ce nom puisse être résolu par ailleurs sur internet (donc enregistré sur un DNS ou … dans un fichier « hosts » comme vous l'avez fait en début de TP). Cette directive prend tout son intérêt lorsque le serveur Apache héberge plusieurs sites web différents (www.toto.fr et www.titi.com par exemple). Il y aura alors une section différente pour chaque site.

`DocumentRoot /var/www/html`

La directive DocumentRoot est essentielle. Elle spécifie la racine relative du site web. Ici la racine est définie à /var/www/html. Cela signifie que la ressource demandée par l'URL http://www.toto.fr/fichier.txt correspondra réellement au fichier /var/www/html/fichier.txt sur le disque dur de la machine serveur.

En d'autres termes tous les fichiers qui seront situés sous ce dossier racine, seront potentiellement accessibles par un navigateur internet, alors que toutes les racines situées au-dessus de cette racine relative resteront innaccessibles. … Autrement dit, si vous définissez cette racine relative à la racine réelle de votre disque dur, votre disque dur devient intégralement accessible par internet ce qui est proprement … suicidaire !

`DirectoryIndex index.html index.htm index.php`

Cette directive spécifie le nom du fichier à renvoyer par défaut dans un dossier, si le nom du fichier n'est pas précisé dans l'URL. Ainsi, si vous invoquez l'URL http://www.toto.fr/dossier1 , Apache ira chercher en priorité dans ce dossier un fichier **index.html**. Si il ne le trouve pas, il cherchera un fichier **index.htm** et enfin si il ne le trouve toujours pas, un fichier **index.php**. Le premier trouvé sera renvoyé au client.

Enfin, si il ne trouve aucun des trois … il cherchera à afficher le contenu du dossier (comme par exemple ici : http://bascou.perso.univ-pau.fr/M2106 où il n'y a aucun fichier d'index disponible). Mais attention… Apache ne pourra afficher le contenu d'un dossier que si il y est autorisé par la directive « **`Options Indexes`** ».

4.3 Principe des sections

Une section définit une condition limitant la portée des directives qu'elle contient

Un type de section très utilisé est le type <Directory>

Ex :
```
<Directory /var/www/html/images>
        DirectoryIndex toto.html
</Directory>
```

Dans cet exemple, la directive *DirectoryIndex* aura un effet limité au dossier */var/www/html/images* ***ET A SES SOUS DOSSIERS***. Sans la section, elle se serait appliquée à l'intégralité du site web.

Qu'en est-il si cette directive est définie à plusieurs endroits ?

Cette question permet d'introduire une notion ESSENTIELLE :

4.4 Priorité des sections.

Ici encore un exemple sera plus parlant :

`DirectoryIndex index.html` (cette première directive est définie hors section, au niveau global)

```
<Directory /var/www/html/images>
            DirectoryIndex toto.html
</Directory>

<Directory /var/www/html/images/thumb>
            DirectoryIndex titi.html
</Directory>

<Directory /var/www/html/images/thumb>
            DirectoryIndex tutu.html
</Directory>
```

Si le fichier apache.conf (après inclusion de tous les fichiers de conf) contient les sections et directives ci-dessus, Apache2 suivra les règles suivantes :

Scénario 1 :

- les directives dans les sections les plus spécifiques sont prioritaires. Par exemple lors d'un accès au dossier /var/www/html/images/thumb, toutes les directives DirectoryIndex correspondent, (la globale et les trois autres), mais les deux dernières sections sont plus spécifiques.
- En cas d'égalité de priorités de sections, la dernière est retenue (d'où l'importance de l'ordre des fichiers inclus, évoquée plus haut). Donc, pour finir l'exemple, Apache2 retiendra la dernière et cherchera un fichier **tutu.html**.

Scénario 2 :

- Lors d'un accès au dossier /var/www/html/images seules les directives DirectoryIndex globale et celle de la première section correspondent, mais la première section est plus spécifique. Donc Apache2 cherchera un fichier **toto.html**

Scénario 3 :

- Un accès au dossier /var/www/html/images/paysages/exterieur/nuit rentre dans le même cas que ci-dessus. Apache2 cherchera un fichier toto.html

Scénario 4 :

- Lors d'un accès au dossier /var/www/html/photos, seule la directive de niveau global correspond, Apache cherchera donc un fichier **index.html**.

Définition : On parle de « **surcharge** » (« override ») d'une directive lorsque sa valeur initiale est modifiée dans une section plus spécifique. Une directive « AllowOverride » sera présentée plus tard en rapport avec cela.

4.5 Autres types de sections

Si <Directory> est le type de section le plus utilisé, il n'est pas pour autant le seul qui existe. Voici quelques autres exemples que vous risquez de rencontrer :

<IfModule xxx> </IfModule>

Les directives de cette section ne seront prises en compte que si le module xxx a été activé

<IfDefine xxx> </IfDefine>

Les directives de cette section ne seront prises en compte que si la variable d'environnement xxx est définie.

<VirtualHost xxx> </VirtualHost>

Les directives de cette section ne concernent que le serveur défini par xxx (xxx précise l'adresse IP et/ou le port). Cette directive existe par défaut depuis la version 2 d'Apache dans le fichier sites-available/000-default. Chaque site hébergé par le serveur doit être spécifié dans une section VirtualHost différente. Vous travaillerez avec cette directive dès le TP numéro 2.

<Files xxx> </Files>

Les directives de cette section ne seront prises en compte que pour les fichiers spécifiés par xxx (*.gif, …).

<Location xxx></Location>

Les directives de cette section ne s'appliqueront qu'aux URL contenant xxx ("/www/"), …

Et il y en a bien d'autres …

5 Quelques exemples de directives

5.1 Directives concernant la configuration générale du serveur

IMPORTANT: les directives sous Apache2 existent en nombre considérable et la liste évolue à chaque nouvelle version. La documentation en ligne est INDISPENSABLE. Elle peut être consultée à l'adresse suivante:

http://httpd.apache.org/docs/2.4/mod/directives.html

où 2.4 peut être remplacé par le numéro de la version désirée.

Les directives ci-dessous sont expliquées à titre d'information, mais ne seront pas toutes utilisées dans le cadre de ce TP. Sauf besoin particulier, la plupart d'entre elles n'ont généralement pas besoin d'être modifiées.

Beaucoup de ces directives peuvent être « surchargées » (« overrided » = modifiées) dans des sections plus spécifiques

ServerRoot "/etc/apache2", précise l'emplacement du dossier sous lequel seront conservés les fichiers de configuration, d'erreur et de logs du serveur Apache (sauf directives contraires explicites pour les fichiers de log : voir Errorlog, TransfertLog …).

User <utilisateur> : définit l'identité de l'utilisateur sous laquelle Apache2 sera exécuté. Il est largement conseillé de le laisser s'exécuter sous l'identité d'un utilisateur qui n'a aucun droit particulier sur le système. Utiliser "root" est à proscrire catégoriquement, un hacker en profiterait immédiatement pour prendre à distance le contrôle de votre machine. Dans les versions actuelles, l'utilisateur *apache*, qui n'a aucun droit, est automatiquement créé et configuré.

Group <group> : définit le groupe sous lequel Apache2 sera exécuté. Mêmes remarques que ci-dessus.

Listen *80*, indique le port sur lequel Apache2 sera à l'écoute. ***Listen 443** (https)*, sous réserve que le module « mod_ssl » soit installé et configuré, permet de placer Apache2 en écoute des requêtes « http**s** » sur le port 443.

Timeout 1200 , précise le temps maximum (en secondes) pendant lequel le serveur attendra la requête suivante d'un client (ceci sert au maintien d'une connexion dans le cas où le transfert complet des données requises nécessite plusieurs messages TCP/IP).

KeepAlive On, précise si les connexions persistantes sont autorisées (maintien de la connexion TCP avec un même client, pour gagner du temps sur la requête suivante).

MaxKeepAliveRequests 100, nombre maxi de requêtes autorisées dans une même connexion persistante. Après cela, la connexion est automatiquement relâchée.

KeepAliveTimeout 15, temps maximum (en secondes) de maintien d'une connexion persistante, sans réception d'une nouvelle requête. Remarque : ne pas confondre avec « Timeout 1200 »…

MinSpareServers 5

MaxSpareServers 10, fourchette dans lequel le nombre de processus fils en attente de connexion va être maintenu en permanence.

StartServers 5, nombre de processus démarrés au lancement d'apache (ce n'est pas une impression : c'est redondant avec MinSpareServers, c'est juste un peu plus rapide !)

MaxRequestPerChild 30, nombre maxi de requête auxquelles peut répondre un processus fils avant d'être supprimé (et donc relancé …). Cette directive est principalement conçue pour s'affranchir des problèmes de gestion de mémoire que présentent certaines implantations d'Unix… sic ! NGINX n'a plus ce problème !!!

MaxClients 150 : Nombre maximum de processus serveurs pouvant co-exister à un instant donné (nb maxi de clients simultanés autorisés).

DefaultType text/plain, type MIME attribué aux fichiers dont le type n'est pas reconnu.

HostnameLookups Off, active la recherche DNS inverse afin d'enregistrer des noms symboliques de clients dans les fichiers de log. Pour des raisons de performances, cette option est généralement désactivée.

ErrorLog /var/log/httpd/error_log, définit l'emplacement du fichier de log d'erreurs (relativement à *ServerRoot* dans le cas d'un nom de fichier relatif).

TransfertLog /var/log/httpd/transfert_log, définit l'emplacement du fichier de log des réponses effectuées par le serveur (relativement à *ServerRoot* dans le cas d'un nom de fichier relatif).

LogLevel warn, fixe le niveau d'enregistrement d'événements d'erreurs (warn par défaut)

LogFormat "%{Referer}I -> %U" referer, permet de créer des types d'événements et de personnaliser le format des messages enregistrés dans les fichiers de logs.

CustomLog /var/log/httpd/referer_log referer, définit le fichier de log pour chaque type d'événement définit avec la directive *LogFormat*.

ServerSignature On, autorise l'introduction de la signature du serveur (Nom, version, OS, …) dans les pages d'erreurs spontanément créées par Apache. Une petite aide supplémentaire pour les pirates éventuels …

Alias /icons/ "/usr/local/httpd/icons", permet de définir des alias, permettant d'accéder à des dossiers (**et sous-dossiers**) ne se trouvant pas sous la racine relative du site (voir « ServerRoot »). Par exemple, il sera possible d'accéder au fichier `/usr/local/httpd/icons/coucou.gif` par l'URL `/icons/coucou.gif`

ScriptAlias /cgi-bin/ "/home/httpd/cgi-bin/", permet de définir un alias spécifiquement pour un dossier contenant des scripts CGI (programmes exécutables utilisés pour construire des pages dynamiques). Bien entendu une section spécifique devra être configurée pour définir les droits nécessaires pour à ce dossier.

5.2 Directives concernant la configuration de chaque site web

*Depuis Apache2, le site web principal (celui répondant par défaut) n'est pas différencié des autres éventuels sites hébergés (les hôtes virtuels). Sa configuration spécifique se trouve dans le fichier **/etc/apache2/sites-available/000-default.conf**, dans une section <VirtualHost>.*

ServerAdmin *webmaster@site.com*, adresse E-mail du webmaster. Cette adresse est souvent retrouvée dans les pages d'erreurs automatiquement composées par le serveur… attention donc, mettez une adresse valide et pouvant être rendue publique ! (une adresse valide n'est pas nécessaire pour ce TP…).

ServerName *www.site.com*, nom par lequel la machine sera appelée. Ce nom doit bien entendu être valide et l'adresse IP correspondante doit pouvoir être obtenue par le système de résolution de noms (dns, …). Cette directive DOIT apparaître dans chaque section « Virtual Hosts » (idéalement, un fichier par Vhost dans le dossier `/etc/apache2/sites-available`)

*** DocumentRoot *"/home/www/site"***, racine relative du site web. Ce dossier correspondra au dossier accessible par l'URL « http://www.site.com ». Il sera donc impossible de référencer à travers une URL un fichier ou dossier se trouvant au dessus de cet emplacement (sauf si des alias sont définis, voir plus loin).

UserDir *public_html*, directive permettant de configurer les racines relatives des utilisateurs enregistrés sur le système hôte. Un dossier « public_html » placé dans le dossier d'accueil de l'utilisateur « marcel » (par exemple : /home/marcel/public_html) sera accessible par l'URL : http://www.site.com/~marcel. Notez que « public_html » n'est qu'un **exemple** de nom de dossier qui peut être avantageusement remplacé par un nom plus court !

DirectoryIndex *index.html index.htm index.cgi*, permet de définir le (ou les) fichiers qui seront en priorité recherchés dans un dossier, dans le cas où aucun n'a été explicitement indiqué dans l'URL.

AccessFileName *.htaccess*, précise le nom du fichier de configuration individuelle d'accès. Ce fichier peut être placé dans un dossier et modifie (assouplit ou renforce) les droits d'accès par défaut attribués à ce dossier (configurés dans httpd.conf). REMARQUE : un tel fichier ne peut être pris en compte par le serveur que si une autorisation explicite lui a été donnée (voir la directive *AllowOverride*)

La syntaxe élémentaire de ce fichier .htaccess pouvant être positionné dans tout dossier est la suivante :

- **Allow:** la directive allow permet de préciser explicitement la ou les machines autorisées. autorise tout le monde. `Allow from univ-pau.fr` autorise toutes les machines appartenant à cette zone ou sous zones. `Allow from All` autorise tout le monde.
- **Deny:** permet d'interdire l'accès à une machine ou un groupe de machines. `Deny from All` interdit tout accès. `Deny from microsoft.com` interdit l'accès à toute machine de cette zone ou sous zone.
- **Order**, cette directive est importante : c'est elle qui définit la politique d'accès par défaut :
  Order *allow,deny* accepte les machines de toute zone sauf ordre explicite contraire par « deny »
  Order *deny,allow* interdit les machines de toute zone sauf ordre explicite contraire par « allow »
  Order *mutual-failure* ne définit pas de politique par défaut.

5.3 Directives concernant les pages générées par le serveur

Dans le cas ou un client demande une URL correspondant à un dossier qui ne contient pas de fichier d'index (voir « *DirectoryIndex* ») et que le serveur est autorisé à le faire (voir « *Options Indexes* »), Apache construit de lui-même une page commençant par « *Index of /* » qui présente la liste des fichiers et sous dossiers. Certaines directives de configuration permettent d'influer sur l'esthétique de cette page :

FancyIndexing on : permet l'utilisation d'icônes fantaisies livrés avec Apache (ou autres).

IndexIgnore .. *.jpg icons HEADER README : permet d'exclure certains fichiers ou dossiers de l'affichage.

HeaderName header : définit le fichier header.html (ou header) comme tel. Ce fichier sera affiché en début de liste (comme en-tête de page).

ReadmeName readme : définit readme.html (ou readme) comme tel. Ce fichier sera affiché après le header en début de liste.

5.4 Directives particulières …

Quelques directives revêtent un caractère un peu particulier et méritent qu'on s'y attarde :

AccessFileName .htaccess : les fichiers généraux de configuration ne sont modifiables que par l'administrateur du serveur (heureusement d'ailleurs ...). Il est parfois utile de disposer de fichiers de configuration spécifiques à chaque dossier (par exemple un utilisateur, dans le cadre de la gestion de ses pages personnelles, peut vouloir adapter quelque peu la configuration à ses besoins, mais uniquement dans ses propres dossiers).

Apache a prévu la possibilité de rajouter un fichier de configuration dans un dossier, dont le nom est défini par cette directive (ici « .htaccess »), dont le contenu est assimilable à une section <Directory> s'appliquant uniquement au dossier qui le contient.

Remarques :

- La balise <Directory> est implicite et ne doit pas apparaître dans le fichier
- Il est conseillé de faire commencer le nom du fichier par un point (« . », fichier caché) pour qu'il n'apparaisse pas dans la liste des fichiers, lorsque le contenu du dossier est affiché à l'écran.
- Seules les directives autorisée par l'administrateur à être surchargées peuvent être modifiées dans le cadre de ce fichier. Cette autorisation est gérée par la directive suivante :

AllowOverride All ne s'utilise que dans une section <Directory>. Il permet à l'administrateur de définir les directives qui peuvent être surchargées dans un fichier d'accès (ex : *.htaccess)*. Certains arguments possibles sont :

- *All* : toutes les directives peuvent être surchargées (par défaut jusqu'à la version 2.3.8)
- *None*: aucune directive ne peut être surchargée (par défaut depuis la version 2.3.9)
- *AuthConfig*: autorise la surcharge des directives (entre autres) *AuthGroupFile, AuthName, AuthType, AuthUserFile*, *require*…(voir le chapitre plus loin sur l'authentification)
- *AuthUserFile*: autorise la surcharge des directives *AuthName*, *AuthType* et *require*
- *Indexes*: autorise la surcharge des directives *FancyIndexing*, etc … (voir 5.3)
- *Options*: autorise la surcharge de la directive *Options*
- *… (voir http://httpd.apache.org/docs/2.4/mod/core.html#allowoverride)*

Options <liste d'options>, cette directive est utilisée pour définir les fonctionnalités disponibles dans un dossier particulier. Elle est généralement utilisée dans des sections <Directory> ou <Virtual Host> ou dans un fichier d'accès (ex : .htaccess) pour conférer des droits particuliers au dossier ou hôte virtuel concerné.

La liste des fonctionnalités principales est la suivante.

- *None*: aucune fonctionnalité particulière
- *All*: toutes les options sont activées (excepté MultiViews) (c'est la valeur de « Options » défaut).
- *ExecCGI*: exécution de CGI autorisée

- *FollowSymLinks:* le serveur est autorisé à suivre les liens symboliques dans ce dossier
- *Indexes:* si une URL fait référence à ce dossier et qu'aucun DirectoryIndex (index.html, …) n'est présent, le serveur retournera une liste du contenu du dossier au lieu d'un message d'erreur.
- *MultiViews:* les MultiViews sont autorisées.
- *SymLinksIfOwnerMatch:* le serveur est autorisé à suivre les liens symboliques du dossier uniquement si les propriétaires du lien et de la cible sont les mêmes.
  Dans le cadre d'une surcharge, les signes + ou – peuvent être utilisés afin de rajouter ou d'enlever des fonctionnalités courantes au lieu de les substituer.

  Ex : **Options +ExecGCI** permet de rajouter l'autorisation d'exécution de programmes CGI à la valeur en cours de Options.

6 Pages soumises à authentification

De nombreux sites web proposent des zones protégées et accessibles uniquement par mot de passe. Plusieurs techniques différentes existent pour réaliser de telles restrictions d'accès. Ici encore les mêmes mécanismes peuvent être implantés dans une section <Directory> ou dans un fichier d'accès (.htaccess) situé dans le dossier à protéger. L'avantage de la seconde solution est de permettre à un utilisateur autre que le webmaster de modifier lui-même la liste des personnes autorisées.

Les directives essentielles sont les suivantes :

AuthType Basic, indique le type de contrôle d'authentification (*Basic* ou *Digest*). *Basic* fait transiter le mot de passe en clair. Cette méthode n'est à utiliser que dans le cadre d'une connexion https (donc déjà chiffrée). *Digest* permet de faire transiter un mot de passe chiffré et est donc utilisable en http, mais cette méthode est plus délicate à mettre en œuvre. Cette directive est obligatoire pour mettre en place une authentification (nous utiliserons Basic dans le cadre de ce TP , même si ce n'est pas conseillé dans la pratique …).

AuthName *domaine* : précise le « domaine apache » auquel appartient la section ou le dossier protégé (cette notion de domaine n'a rien à voir avec les domaines dns, nis, windows ou autres …). Un internaute pourra accéder à toutes les pages protégées appartenant à un même domaine, en ne s'étant authentifié qu'une seule fois pour la première page. Un domaine doit obligatoirement être spécifié, pour chaque ressource protégée (section ou dossier).

AuthGroupFile *groupfile*, permet de définir un fichier contenant des noms de groupe d'utilisateurs ainsi que les utilisateurs membres. Cette directive est facultative. Exemple du format de *groupfile* :

groupe1 : marcel louis alfred
groupe2 : marcel louis hubert

AuthUserFile *userfile*, permet de définir le nom d'un fichier d'utilisateurs autorisés ainsi que leurs mots de passe. ATTENTION : ce fichier n'est pas généré par édition manuelle, mais par l'utilisation de la commande unix *htpasswd* (jetez un coup d'œil sur le « man » unix).

Require fixe les utilisateurs autorisés à accéder. Les arguments de cette directive obligatoire peuvent être :

- *user marcel louis hubert*, pour autoriser une liste d'utilisateurs référencés dans *userfile*
- *group groupe1*, pour autoriser une liste de groupes définis dans *groupfile*
- *valid-user*, pour autoriser tous les utilisateurs référencés dans le fichier *userfile*

REMARQUE TRES IMPORTANTE : Si les fichiers *userfile* et *groupfile* sont indiqués en relatif, ils le sont relativement au dossier racine de configuration, défini par la directive « ServerRoot ». Pour les mettre dans un autre dossier ailleurs, il suffit de préciser leurs noms absolus.

Ex : **userfile users.pwd** fait référence à **/etc/apache2/users.pwd** (avec: « ServerRoot /etc/apache2 »**)**

7 Les hôtes virtuels ("Virtual Hosts" ou VH)

Les premières versions d'Apache ne permettaient pas d'héberger efficacement plusieurs sites sur un même serveur. La version 2 a révolutionné ce concept avec les « Virtual Hosts », à tel point que même si le serveur n'héberge qu'un seul site, il sera quand même considéré comme un site virtuel.

Les hôtes virtuels (i.e les différents sites) peuvent être différenciés :

- Selon leur adresse IP (il faut alors configurer plusieurs adresses IP au serveur)
- Selon le type d'adresse IP (v4 ou v6)
- Selon le port (il faut pour cela que Apache écoute sur plusieurs ports)
- Selon le nom utilisé dans l'URL (on parle de NVH pour Named VH, c'est le plus courant)
- Selon une combinaison des points précédents

Apache disposera dans ses fichiers de configuration d'une section <VirtualHost xxx> pour chaque cas à envisager, chacune disposant d'une configuration différente. A l'intérieur de chaque section, les directives essentiellement

nécessaires sont: *DocumentRoot* et (pour les NVH) *ServerName*. Dans la pratique, il peut être intéressant d'utiliser *ServerAdmin* (un mail différent pour chaque site), ainsi que les directives concernant les fichiers de logs (pour les séparer des logs des autres sites).

Même si ce n'est pas obligatoire, il est LARGEMENT RECOMMANDÉ de créer un fichier séparé (dans le dossier « site-available »), pour chaque site virtuel. Il sera alors plus simple grâce aux commandes **a2ensite** et **a2dissite** d'activer ou désactiver un site en particulier sans impacter les autres.

Voici quelques exemples de cette balise de section :

<VirtualHost *:*> </VirtualHost> : site concerné quelle que soit l'adresse IP utilisée, quel que soit le port utilisé.

<VirtualHost 12.12.12.12:*> </VirtualHost> site concerné par les requêtes adressées à l'adresse IP 12.12.12.12, sur n'importe quel port.

<VirtualHost *:8080> </VirtualHost> : site concerné par les requête adressées au port 8080, quelle que soit l'adresse IP utilisée.

<VirtualHost [FE80::12:12]:8080> </VirtualHost> site concerné par les requêtes adressées uniquement sur l'adresse IPv6 FE80::12:12 et uniquement sur le port 8080

… etc …

8 Travaux pratiques :

REMARQUES PRELIMINAIRES IMPORTANTES

- Pour la page d'accueil de votre site web, pour gagner du temps, un fichier html « index.html » est disponible sous /var/www/html. Il est fourni en standard avec les distributions actuelles d'apache2. Vous pouvez le copier, le réutiliser et le modifier à loisir. Vous pouvez le visualiser pour voir à quoi il correspond.

- Il se peut que vous constatiez parfois un fonctionnement « anormal » au fur et à mesure de vos manipulations. N'oubliez pas que les navigateurs web ont une fâcheuse tendance à aller puiser dans leur cache, les pages qu'ils ont déjà consultées. Pour être sûr de visualiser les dernières modifications apportées, cliquez sur « rafraîchir » (reload), ou « majuscule+rafraîchir » (shift+reload), voire carrément effacer votre historique …

- De même pour les sections protégées par mot de passe : un mot de passe est souvent mémorisé par le client web. Il est donc probable qu'il ne vous le demande plus, si vous l'avez correctement fourni une première fois. La solution la plus sûre consiste à changer le mot de passe sur le serveur !

- Bien évidemment, pour ce TP, aucun serveur proxy ne doit être configuré !

- **Consultez régulièrement les fichiers de logs (surtout ceux d'erreurs dans /var/log/apache2/error_log). Ils vous permettront de gagner un temps considérable dans la phase de mise au point.**

- Si au bout de 30 minutes (en supposant que vous avez déjà lu le sujet) vous n'avez pas encore fait valider le 1er point, appelez l'enseignant pour vous aider, sinon vous y serez encore au bout de trois heures.

8.1 Configuration générale

- Mettez en place la configuration de votre environnement telle qu'elle est expliquée au paragraphe 1.2 et inscrivez ci-dessous vos trois noms fqdn :

adehu.iutmdm.univ-pau.fr

adehu2.iutmdm.univ-pau.fr

adehu3.iutmdm.univ-pau.fr

- Configurez le serveur Apache pour mettre en place le site répondant au nom fqdn de votre machine « XX.iutmdm.univ-pau.fr », sur le port standard (80). Testez son fonctionnement avec une page d'accueil quelconque (il n'y a pas grand-chose à changer aux fichiers de config pour faire marcher ceci…). Quelles modifications avez-vous dû effectuer et où ?

```
$ pluma admin:///etc/hosts
127.0.0.1 adehu.iutmdm.univ-pau.fr
127.0.0.1 adehu2.iutmdm.univ-pau.fr
127.0.0.1 adehu3.iutmdm.univ-pau.fr
```

- Modifiez la configuration pour définir la racine du site sur /home/web (au lieu de /var/www/html) dans laquelle vous installerez une page d'accueil.
  Remarque1 : Faites attention aux droits : le serveur apache s'exécute en tant que « user » du groupe « group » (selon les directives correspondantes dans le fichier de conf). Il doit néanmoins pouvoir « passer » par les dossiers /home et /home/web et pouvoir lire les fichiers de /home/web. A cet effet vous préciserez les droits Unix minimums que vous êtes obligés de positionner sur ces ressources.
  Remarque2 : ATTENTION : il faut également mettre en correspondance dans les fichiers de configuration certaines directives qui attribuent des droits d'accès privilégiés à /var/www/html (notamment dans la section « Directory » du fichier de config principal). Résumez ci-dessous les manipulations effectuées :

```
$ pluma admin:///etc/apache2/apache2.conf
```

```
$ pluma admin:///etc/apache2/sites-available/000-default.conf
DocumentRoot /home/web

# mkdir /home/web
# cp /var/www/html/index.html /home/web/
# systemctl restart apache2
```

- Modifiez la configuration pour définir « welcome.html » ou « welcome.htm » comme page d'accueil par défaut. Renommez le fichier index.html en welcome.html et vérifiez que cela marche. Qu'avez-vous dû changer ?

```
$ pluma admin:///etc/apache2/sites-available/000-default.conf
DirectoryIndex welcome.htm

# systemctl restart apache2
```

- Modifiez la configuration pour que le serveur réponde également sur le port 8080. Testez le en essayant l'URL « XX.iutmdm.univ-pau.fr :8080 ». Quelle modification avez-vous faite ?

```
$ pluma admin:///etc/apache2/ports.conf
Listen 8080

# systemctl restart apache2
```

8.2 Pages personnelles

- En tant qu'utilisateur « jules » qui disposez d'un compte sur la machine serveur, vous souhaitez installer votre page personnelle (« **perso.html** ») dans le dossier /home/jules/www/.
  Testez son fonctionnement : elle devra répondre à l'URL : « **XX.iutmdm.univ-pau.fr/~jules/perso.html** » tout d'abord, puis à l'URL : « **XX.iutmdm.univ-pau.fr/~jules/** » après amélioration de la configuration.
  Pour cela il vous faudra créer un utilisateur « jules », installer un module Apache additionnel (le module « userdir », vous documenter un peu sur son fonctionnement et réfléchir un minimum. Expliquez et justifiez ci-dessous les manipulations que vous avez effectuées:

```
# a2enmod userdir
# mkdir /home/xeylou/public_html/
# echo '1' > /home/xeylou/public_html/welcome.htm
# systemctl restart apache2
```

```
note: we configured the DirectoryIndex as
welcome.htm, so if we create a file w/
this name in the user's public_html file
it will redirect to it when accessing
adehu.iutmdm.univ-pau.fr/~user

if we created index.html or another file
w/ different name, it will prompt the apache
directory searching
only works with os users
```

8.3 Sécurisation d'accès

- Créez le dossier /home/web/private. En modifiant la configuration globale, créez une section concernant ce dossier dans laquelle vous positionnerez les directives nécessaires pour restreindre l'accès à trois utilisateurs privilégiés riri, fifi et loulou auxquels vous attribuerez des mots de passe. Installez une page d'accueil dans ce dossier et testez l'efficacité de votre protection. Vous devrez pour cela créer une section spécifique au dossier dans le fichier de config du site, y indiquer les bonnes directives et créer le fichier contenant les mots de passe.

```
$ pluma admin:///etc/apache2/apache2.conf<Direc
```

```
$ pluma admin:///etc/apache2/sites-available/000-default.conf
DocumentRoot /home/web/privateDirectoryIndex index.html

# mkdir /home/web/private

# mkdir /usr/local/apache/
# mkdir /usr/local/apache/conf/
# htpasswd /usr/local/apache/conf/auth.users
# echo '5' > /home/web/private/index.html
```

https://www.feistyduck.com/library/apache-security/online/apachesc-CHP-7.html

- Jules souhaite maintenant se créer un dossier à accès sécurisé « secret » qui correspondra à l'URL « XX.iutmdm.univ-pau.fr/~jules/secret ». N'étant pas administrateur, il est donc obligé de passer par un fichier d'accès (.htaccess). Mettez en place ce fichier contenant les directives nécessaires (choisissez un nouveau nom de domaine et un nouveau fichier de mot de passe caché dans le dossier /home/jules). Faites en sorte que cela fonctionne. Indiquez ci-dessous le contenu du fichier /home/jules/www/secret/.htaccess

nano /etc/apache2/sites-available/default-ssl.conf

$ mkdir ~/www$ mkdir ~/www/sec

nano /etc/apache2/sites-available/apache2.conf

.htaccess is defined at the bottom of apache2.conf, apache will look for one for each directory it can access

- Créez le dossier /home/web/private2. Mettez en place un système de protection local par fichier d'accès en utilisant cette fois un fichier d'accès qui devra s'appeler « .passport » (au lieu de «.htaccess ») que vous créerez dans le dossier private2. Précisez un domaine différent pour ne pas interférer avec la question précédente. Installez une page d'accueil dans ce nouveau dossier et testez l'efficacité de votre protection. Attention, n'oubliez pas que pour que le fichier « .passport » soit pris en compte, il faut que l'administrateur en ait donné l'autorisation (directive « AllowOveride All », dans la section correspondante au dossier /home/web/private2 des fichiers de configuration d'Apache2). Voir le paragraphe 5.4. Précisez vos manipulations :

8.4 Installation de « Named Virtual Hosts »

Remarque préliminaire: Les trois noms XX.iutmdm.univ-pau.fr, XX2.iutmdm.univ-pau.fr et XX3.iutmdm.univ-pau.fr correspondent à la même adresse IP : celle de votre serveur.

- Créez les dossiers /home/web/site2 et /home/web/site3. Installez une page d'accueil d'une couleur de fond différente dans chacun. Configurez alors le serveur Apache pour qu'il reconnaisse les deux NVH : site 2 sur « XX2.iutmdm.univ-pau.fr » et site3 sur « XX3.iutmdm.univ-pau.fr ».
  Vérifiez que lors de l'interrogation par un navigateur, les trois pages sont différentes selon que l'on demande le site « principal », le VH2 ou le VH3.
  Pour faire cela correctement (i.e. proprement), il vaut mieux créer un nouveau fichier de configuration dans « sites-available » pour chaque VHost. Bien sûr, n'oubliez pas de créer le lien symbolique correspondant dans « sites-enables » grâce à la commande « a2ensite ». Retranscrivez vos manipulations ci-dessous :

- Créez les dossiers /home/web/site2-8080 et /home/web/site3-8080. Installez une page d'accueil différente dans chacun.
  Configurez Apache pour qu'il prenne en compte deux NVH supplémentaire sur les ports 8080.
  Au final, Apache devra renvoyer une page web différente selon qu'il sera accédé par l'intermédiaire des l'URLs :
  - « XX.iutmdm.univ-pau.fr »
  - « XX.iutmdm.univ-pau.fr :7000 »
  - « XX2.iutmdm.univ-pau.fr »
  - « XX.iutmdm.univ-pau.fr :7000 »
  - « XX3.iutmdm.univ-pau.fr »
  - « XX3.iutmdm.univ-pau.fr :7000 »
- Normalement, les six URLs précédentes doivent vous renvoyer six pages différentes …

Copiez ci-dessous le contenu de tous vos fichiers du dossier « sites-available » :