Cours/TP n° 1
Installation et configuration d'un serveur web

1 Présentation Générale

Les cours Web de cette année auront pour objectifs principaux de vous faire découvrir ou redécouvrir un large éventail de technologies utilisées dans les différentes étapes de la conception d'un site web. Pour ce faire, nous nous placerons dans le cadre d'un particulier (ou d'une PME) qui souhaite héberger lui-même son site web ainsi que différents outils largement utilisé de nos jours. Pour cela, le cours sera organisé selon 3 parties. Les 3 parties seront les suivantes :

- Installation et Configuration d'un serveur web et d'autres outils,
- Présentation des technologies dites « coté client »,
- Présentation des technologies dites « coté serveur »,

Mais commençons immédiatement avec notre premier TP.

2 Introduction

Les objectifs de ce TP sont de vous montrer comment installer et configurer un serveur web. Nous verrons dans un premier temps la configuration d'un serveur Apache sur un système *Debian 8 – server* puis la configuration de divers outils pour terminer par la configuration du serveur IIS de Microsoft sur un système *Windows*.

2.1 Le rendu

Vous devrez en fin de séance rendre un compte rendu de votre TP. Ce compte rendu devra **OBLIGATOIREMENT** respecter les contraintes suivantes :

- Etre envoyez par email à votre enseignant.
- Avoir le sujet suivant : *[2015][LPSIL][IDSE][TP1]nom1_Prenom1 – nom2_Prenom2* où nom1_Prenom1 et *nom2_Prenom2* les noms et prénoms du binôme.
- Etre reçu par votre enseignant le jour du cours/TP.
- Contenir dans le corps de l'email les questions que vous pourriez avoir (si vous souhaitez avoir une réponse rapide car les comptes rendu ne sont pas évalués immédiatement).
- Contenir en attachement 1 seul fichier .zip, .7z ou .tar.gz contenant lui-même 1 fichier de compte rendu (au format pdf, rtf, doc, docx ou txt) ainsi que les divers fichiers (fichiers de configuration, pages (x)html, CSS, php…) écrits dans le TP. Ne pas joindre de fichiers exécutables.

2.2 Serveur HTTP

Un ordinateur sur lequel fonctionne un serveur HTTP est appelé serveur web. Dans la suite du cours nous utiliserons indifféremment le terme « serveur web » pour désigner le serveur HTTP (le logiciel) lui-même. Même si nous nous limiterons à deux exemples de serveurs pour nos TPs, il est bon de savoir qu'il existe d'autres serveurs http que ceux que nous utiliserons. Voici une petite liste des principaux serveurs HTTP du marché :

- Apache HTTP Server de la *Apache Software Foundation,*
- Internet Information Services (IIS) de Microsoft,
- Oracle Java System Web Server de Oracle (anciennement Sun Microsystems),
- Zeus Web Server de Zeus Technology ;

- Lighttpd de Jan Kneschke ;
- Nginx d'Igor Sysoev ;
- Cherokee de Alvaro López Ortega.

2.3 Machine virtuelle

Pour réaliser ce cours/TP nous allons vous fournir par groupes de 2 étudiants un serveur (si vous le souhaitez, vous pouvez faire les cours/TP seul, dans la limite des machines virtuelles disponibles). Vous n'aurez pas accès physique à cette machine mais vous allez pouvoir vous y connecter par ssh et via les protocoles http et https. Pour cela vous aurez besoin de différents outils (libre à vous de choisir les vôtres, mais dans ce cas vous devez savoir les utiliser):

- Un client ssh pour une connexion sécurisé vers le serveur, nous vous conseillons Putty ;
- Un logiciel pour faire des transferts de fichiers vers le serveur, nous vous conseillons WinSCP ;
- Un navigateur web récent, nous vous conseillons Opera 31 ;
- Un éditeur de textes orienté pour le développement web (WebExpert pour ceux qui ont une licence) ou Notepad++

Commencez par installer vos outils et allez demander un serveur au responsable du cours. **Attention, vous ne devez utiliser ce serveur que pour le cours web.** Toutes personnes utilisant ce serveur à d'autres fins, s'en verra supprimé l'accès.

2.4 Informations sur les systèmes

L'ensemble des serveurs virtuels que nous vous proposons disposent d'un utilisateur déjà créé. Ce compte peut exécuter des commandes avec les droits d'administration sur le système via la commande *sudo*.
Login : student Pass : student2015!
Pensez à immédiatement changer le mot de passe de ce compte. Attention à ne pas perdre votre nouveau mot de passe sans quoi vous ne pourrez pas accéder à votre serveur par la suite.
Rappels de quelques commandes linux:

- *sudo* : permet d'exécuter une commande avec le privilège d'administration.
- *passwd* : permet de changer son mot de passe.
- *loadkeys fr* : passe le clavier en français (fr) (nécessite les droits d'administration du système).
- *nano* : un éditeur de texte en ligne de commande.
- *vi* : un autre éditeur de texte en ligne de commande.

Attention, ne pas arrêter le serveur virtuel, car vous ne pourrez pas le redémarrer.

3 Installation et configuration d'un serveur Apache sous Linux (Obligatoire)

Connectez-vous via ssh sur votre serveur en utilisant l'adresse IP et le compte que vous a fourni le responsable du TP.
Si vous avez un clavier azerty et que le système n'est pas configuré pour, exécutez la commande suivante pour que le système passe sur un clavier français.

```
> loadkeys fr
```

3.1 Installation du serveur

Pour simplifier l'installation et ne pas perdre de temps, nous allons utiliser le système de package de Debian. Si vous souhaitez (en dehors du TP) installer complètement le serveur Apache de manière manuelle, vous trouverez ce dont vous avez besoin à l'adresse suivante : www.apache.org/dist/httpd/ ainsi que la documentation. La dernière version au moment de la rédaction de ce TPs est Apache HTTP Server 2.4.16 ou la 2.2.31 (si on reste sur la branche 2.2).

Commencez par mettre à jour votre système. Debian utilise le système de paquets dpkg / apt.

```
> apt-get update
> apt-get upgrade
ou
> aptitude update
> aptitude safe-upgrade
```

Pour faire simple dans un premier temps, nous allons installer notre serveur apache http en utilisant les paquets Debian. Nous verrons plus tard, comment faire cette installation à la main.

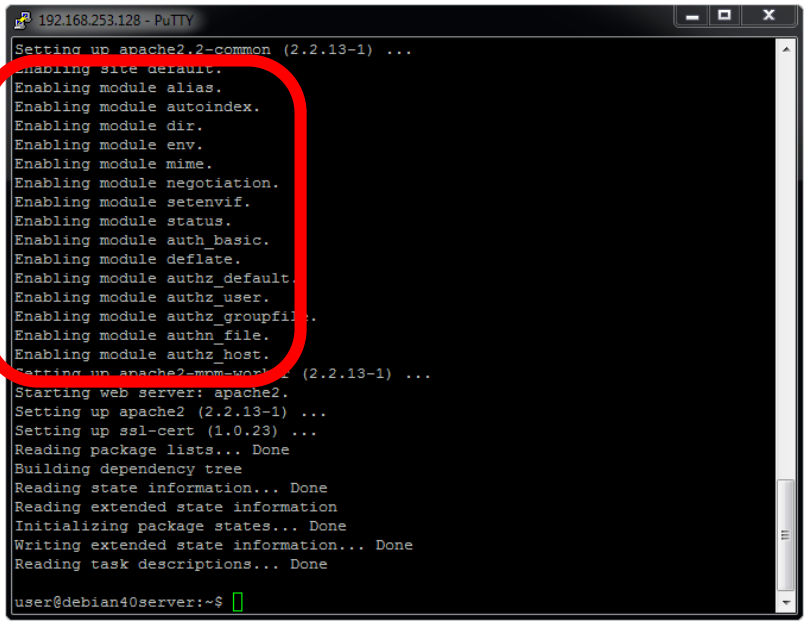
Nous pouvons commencer à installer le serveur http Apache. Nous utiliserons ici la version 2.4 du serveur. Il est possible que la version déployée par le système de paquets soit plus ancienne que celle disponible sur le site de la fondation apache. Nous verrons dans la suite de ce cours/TP comment installer directement la version de notre choix.

```
> apt-get install apache2
ou
> aptitude install apache2
```

Regardez la liste des packages installés ou mis à jour. Certains packages déjà installés seront également mis à jour. Il est important de bien comprendre que cela peut avoir un impact sur votre système.

Si aucune n'erreur n'est signalée, l'installation d'Apache est terminée.

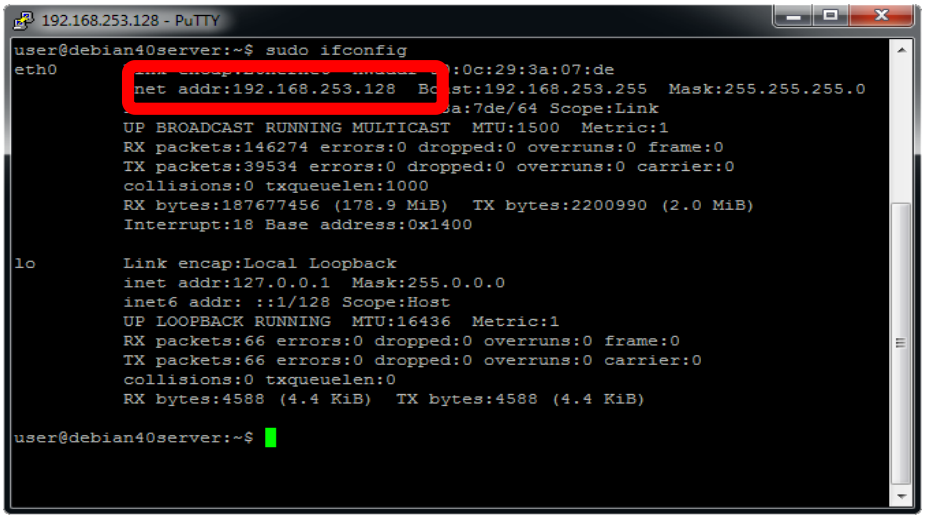
Avant toute chose, notez la liste des modules activés par l'installation par défaut.



Quelle est la version de HTTP apache 2 qui a été installé sur votre système ?

Maintenant vérifiez si votre serveur http fonctionne. Vous devriez pourvoir faire afficher la premier page web en allant à l'adresse http://xxx.xxx.xxx.xxx/ (IP de votre serveur). Pour connaitre l'adresse de votre serveur (si vous l'avez déjà oublié), tapez la commande suivante :

```
> ifconfig
```



Bravo, vous venez de finir l'installation de votre serveur http apache sous linux. Maintenant nous allons voir comment configurer celui-ci pour qu'il réponde correctement à nos besoins.

3.1.1 Arrêt et redémarrage

Tout au long de ce cours/TP nous aurons besoin d'arrêter et de redémarrer notre serveur http apache pour tester ces différentes fonctionnalités. Voici donc rapidement les différentes commandes permettant de faire cela. Attention, il est possible que vous ayez besoin des droits d'administrateur pour ces commandes. Dans ce cas, pensez à vous connecter sur le compte root.

Démarrage normal

Démarre normalement le processus daemon *httpd* (apache2 dans notre cas).

```
> apache2ctl -k start
```

Arrêter immédiatement

L'envoi du signal stop au processus parent induit chez celui-ci une tentative immédiate de tuer tous ses processus enfants. Cela peut durer plusieurs secondes. Après cela, le processus parent lui-même se termine. Toutes les requêtes en cours sont terminées, et plus aucune autre n'est traitée.

```
> apache2ctl -k stop
```

Arrêt en douceur

L'envoi du signal graceful-stop au processus parent lui fait aviser les processus enfants de s'arrêter après le traitement de leur requête en cours (ou de s'arrêter immédiatement s'ils n'ont plus de requête à traiter). Le processus parent va alors supprimer son fichier *PidFile* et cesser l'écoute de tous ses ports. Le processus parent va continuer à s'exécuter, et va surveiller les processus enfants qui ont encore des requêtes à traiter. Lorsque tous les processus enfants ont terminé leurs traitements et se sont arrêtés ou lorsque le délai spécifié par la directive GracefulShutdownTimeout a été atteint, le processus parent s'arrêtera à son tour. Si ce délai est atteint, tout processus enfant encore en cours d'exécution se verra envoyer le signal stop afin de le forcer à s'arrêter.

```
> apache2ctl -k graceful-stop
```

Redémarrer immédiatement

L'envoi du signal restart au processus parent lui fait tuer ses processus enfants comme pour le signal stop, mais le processus parent ne se termine pas. Il relit ses fichiers de configuration, et réouvre ses fichiers de log. Puis il donne naissance à un nouveau jeu de processus enfants et continue de traiter les requêtes.

```
> apache2ctl -k restart
```

Redémarrage en douceur

L'envoi du signal graceful au processus parent lui fait envoyer aux processus enfants l'ordre de se terminer une fois leur requête courante traitée (ou de se terminer immédiatement s'ils n'ont plus rien à traiter). Le processus parent relit ses fichiers de configuration et rouvre ses fichiers de log. Chaque fois qu'un enfant s'éteint, le processus parent le remplace par un processus enfant de la nouvelle génération de la configuration, et celui-ci commence immédiatement à traiter les nouvelles requêtes.

```
> apache2ctl -k graceful
```

Cas particulier sur Debian

Sur un système Debian, il est conseillé d'utiliser les scripts mise en place sur ce système pour gérer les différentes applications. Nous n'utiliserons donc pas les commandes présentés ci-dessus mais la commande suivante :

```
> /etc/init.d/apache2 [start|restart|stop|reload|force-reload|status]
ou
> service apache2 [start|restart|stop|reload|force-reload|status]
```

L'utilisation de ce script garanti le bon fonctionnement du système et des commandes (vérifications des variables d'environnements, …).

Vérifier ce que fait ce script et regardez s'il s'appuie sur les commandes présentées plus haut :

1. En comparant l'effet des commandes standards et celles du script. Vous pourrez par exemple utiliser la combinaison des commandes ps et grep pour vérifier l'arrêt et le redémarrage du serveur.
2. En regardant le code du script directement.

3.2 Configuration du serveur

Par défaut, les versions d'Apache n'utilisaient qu'un seul fichier de configuration nommé *httpd.conf* et situé dans le répertoire */etc/apache2/*. Dans la version que nous installerons (et toutes celles basées sur un système Debian), les informations du fichier *httpd.conf* sont réparties dans plusieurs fichiers. Il est possible que ce fichier n'existe même plus sur votre système (en fonction de la version installée).

Si le fichier *httpd.conf* est présent, vérifiez qu'il est bien vide à l'aide de la commande suivante :

```
> ls /etc/apache2/
> cat /etc/apache2/httpd.conf
```

De manière générale, il est conseillé de faire une copie de sauvegarde des fichiers avant toutes modifications. Par la suite, dans ce cours/TP, nous ne vous rappellerons pas à chaque fois de faire une sauvegarde des fichiers que nous modifierons. Pensez-y ou vous risquez de devoir reprendre le TP depuis le début.

3.2.1 Syntaxe des fichiers de configuration

Les fichiers de configurations (*httpd.conf, apache2.conf, …)* contiennent deux types d'informations : des commentaires et des directives pour le serveur. Les lignes commençant par le caractère # sont traitées comme des lignes de commentaires; ces commentaires n'ont aucune utilité pour le logiciel du serveur, mais ils servent de documentation pour l'administrateur du serveur. Vous pouvez ajouter autant de commentaires que vous voulez; le

serveur ignore simplement tous les commentaires quand il fait l'analyse syntaxique du fichier. Les commentaires ne doivent pas être inclus dans une ligne après une directive de configuration. Les lignes vides et les espaces précédant une directive sont ignorés; vous pouvez par conséquent indenter les directives afin d'améliorer la lisibilité.
Vérifiez si vos fichiers de configuration contiennent des erreurs de syntaxe avec la commande suivante :

```
> apache2ctl -t
```

3.2.2 Modules

Apache est un serveur modulaire. Ceci implique que seules les fonctionnalités les plus courantes sont incluses dans le serveur de base. Les fonctionnalités étendues sont fournies à l'aide de modules qui peuvent être chargés dans *Apache*. Par défaut, un jeu de modules de base est inclus dans le serveur à la compilation. Si le serveur est compilé de façon à utiliser les modules chargés dynamiquement, alors les modules peuvent être compilés séparément et chargés à n'importe quel moment à l'aide de la directive *LoadModule*. Dans le cas contraire, Apache doit être recompilé pour ajouter ou supprimer des modules. Les directives de configuration peuvent être incluses de manière conditionnelle selon la présence ou l'absence d'un module particulier en les plaçant dans un bloc *<IfModule>*.
Vérifiez les modules qui ont été compilés avec le serveur avec la commande suivante :

```
> apache2ctl -l
```

Rendez-vous dans le répertoire */etc/apache2/*.

3.2.3 Configuration d'apache2.conf

Ouvrez le fichier avec votre éditeur de texte préféré et parcourrez le fichier. Par exemple, utiliser la commande suivante pour l'ouvrir avec vi.

```
> vi apache2.conf
```

Voici un exemple de fichier *apache.conf* auquel nous avons retiré, en grande partie, les commentaires pour des questions de lisibilité.

```
# Global configuration
#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
# Do NOT add a slash at the end of the directory path.
#ServerRoot "/etc/apache2"

# The accept serialization lock file MUST BE STORED ON A LOCAL DISK.
Mutex file:${APACHE_LOCK_DIR} default

# PidFile: The file in which the server should record its process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
PidFile ${APACHE_PID_FILE}

# Timeout: The number of seconds before receives and sends time out.
Timeout 300

# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
KeepAlive On
```

```
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
KeepAliveTimeout 5

# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
HostnameLookups Off

# ErrorLog: The location of the error log file.
ErrorLog ${APACHE_LOG_DIR}/error.log

# LogLevel: Control the number of messages logged to the error_log.
LogLevel warn

# Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

# Include list of ports to listen on and which to use for name based vhosts
Include ports.conf

# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

```
# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
AccessFileName .htaccess

# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
<FilesMatch "^\.ht">
        Require all denied
</FilesMatch>

# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\""
vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include generic snippets of statements
IncludeOptional conf.d/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
```

Examinez votre propre fichier de configuration et essayez de comprendre par vous-même (sans lire la suite de ce cours ni chercher des informations sur le net) l'ensemble des directives présentes. Quand vous pensez avoir compris, essayez de répondre aux questions ci-dessous.

- Vérifiez si le numéro du processus linux du démon *httpd* (*apache2* dans notre cas) contenu dans le fichier *apache.pid* correspond bien au premier processus apache2.
- Combien y a-t-il de processus apache2 actuellement fonctionnel ?
- Que se passe-t-il sur les numéros des processus apache2 après l'exécution d'une commande de redémarrage ? Vous vérifiez notamment si les commandes `apache2ctl -k restart` et `/etc/init.d/apache2 restart` ont le même effet sur le serveur apache httpd.
- Vérifiez dans le fichier le nom d'utilisateur et du groupe d'apache.
- Que contiennent les répertoires */etc/apache/mods-available, /etc/apache/mods-enabled, /etc/apache/ sites - available et /etc/apache/sites-enabled* ?

Nous allons maintenant utiliser la directive *DirectoryIndex* pour spécifier quel fichier doit être lu si aucun n'est défini par la requête http. Si le module *dir* n'a pas été installé au début, activez le module « dir » à l'aide d'une des méthodes suivantes (classées de la moins bonne à la plus correcte)

- copiez les fichiers d*ir.conf* et d*ir.load* situés dans */etc/apache2/mods-available/*dans */etc/apache2/mods-enabled/*.
- Faire un lien symbolique des fichiers d*ir.conf* et d*ir.load* situés dans */etc/apache2/mods-available/*vers */etc/apache2/mods-enabled/*.
- Utiliser la commande apache2 « *a2enmod* » pour activer le module (la commande a2dismod permet de désactiver un module):

```
> a2enmod dir
```

Modifier la ligne suivante dans votre fichier *dir.conf* puis redémarrer le serveur http.

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

De manière générale, si on veut qu'une modification effectuée dans un des fichiers de */etc/apache2* soit prise en compte, il faut redémarrer le serveur web.

Consulter l'adresse suivante http://10.0.2.xxx/ (IP de votre serveur) depuis votre navigateur préféré.

Puisque aucune page *index.htm* n'existe de */var/www/html/* (répertoire actuellement défini pas défaut), c'est le fichier *index.html* qui est lu. Créez le fichier *index.htm* suivant dans */var/www/html/* :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xml:lang="en" lang="en" xmlns="http://www.w3.org/1999/xhtml">
	<head>
		<title>Ma page index.htm</title>
	</head>
	<body>
		<p>Ma page index.htm</p>
	</body>
</html>
```

Sans redémarrer le serveur, rechargez votre page web (reconsulter l'adresse suivante http://xxx.xxx.xxx.xxx/ (IP de votre serveur) depuis votre navigateur préféré). Normalement c'est votre page *index.htm* qui devrait être affichée. Attention, des fois la page est en cache dans votre navigateur, donc il faut forcer le rechargement.

De manière générale, si on fait une modification dans l'espace web (*/var/www*), on n'a pas besoin de redémarrer le serveur web pour qu'elle soit prise en compte.

Maintenant nous allons voir comment créer des répertoires spécifiques pour chaque utilisateur du système.

Si vous n'avez pas d'utilisateur autre que root, vous pouvez en créer un avec les commandes suivantes

```
> useradd -m -d /home/student/ -s /bin/bash student
> passwd student
```

Par exemple, notre utilisateur *student* aura un espace privé (*public_html* par exemple) sur son *homedir* (*/home/student/*) qui sera accessible via l'url http://xxx.xxx.xxx.xxx/~student / (avec l'IP de votre serveur).
Commencez par créer un répertoire *public_html* dans le *homedir* de *student* (attention, de bien faire cela depuis le compte *student* et non le compte *root*).

Pour passer facilement sous le compte student, utilisez la commande suivante :

```
> su student
```

Pour aller dans le *homedir* de *student* :

```
> cd
```

Pour revenir au compte *root* :

```
> exit
```

Revenez sous le compte *root*. Il vous reste à installer le module *userdir* (de la même manière que pour le module *dir*) et enfin de redémarrer le serveur http. Le paramétrage se fait dans le fichier *userdir.conf*.

```
UserDir public_html
```

Affichez la page suivante pour vérifier si ça marche : http://xxx.xxx.xxx.xxx/~student/.

Vérifiez la présence de ces lignes à la fin du fichier *apache2.conf* car nous en aurons besoins dans la suite de ce cours/TPs.

```
# Include the virtual host configurations:
Include sites-enabled/
```

4 Le protocole http (Obligatoire)

Ce cours/TP a pour but de vous faire tester simplement les requêtes et réponses du protocole http. Exécutez depuis Windows ou linux la commande suivante pour vous connecter à votre serveur http.

```
telnet xxx.xxx.xxx.xxx 80
```

4.1 Première requête

Maintenant nous allons envoyer notre première requête http. Envoyez la commande suivante et vérifiez si vous obtenez bien la page web correspondante.

```
GET
```

Comme la requête n'est pas valide nous devrions avoir une erreur. Mais le serveur nous retourne normalement la page web sans rien d'autre. Ecrivons maintenant une requête http valide la plus simple possible. Pour rappel, une requête doit avoir la forme suivante (les informations en [] sont facultatives) :

```
<Méthode> <URI> HTTP/<Version>
[<Champ d'entête>: <Valeur>]
[<tab><Suite Valeur si >1024>]
Ligne_vide (CRLF)
```

La requête la plus simple est donc la suivante (*n'oubliez pas la ligne vide*):

```
GET / HTTP/1.0

```

Vérifiez que vous obtenez bien quelque chose comme cela pour un fichier xhtml:

```
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 13:18:06 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 51
Connection: close
Content-Type: application/xhtml+xml

<html><body><h1>It works! xhtml</h1></body></html>
```

Ou comme cela pour un fichier html

```
HTTP/1.1 200 OK
```

```
Date: Tue, 20 Aug 2013 13:18:06 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 51
Connection: close
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>
```

4.2 Test des codes d'erreurs

Le but est de tester les différents codes d'erreurs utilisés dans le protocole HTTP.

4.2.1 Erreur 404

Commençons par le code 404 : page no found. Ce code est retourné quand la page demandée n'existe pas. Pour vérifier cela, envoyez la requête suivante en vérifiant que vous n'avez pas le fichier tyuio à la racine de votre serveur

```
GET /tyuio http/1.0
```

Normalement vous deviez avoir le résultat suivant :

```
HTTP/1.1 404 Not Found
Date: Tue, 09 Sep 2008 15:41:22 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Vary: Accept-Encoding
Content-Length: 318
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /tyuio was not found on this server.</p>
<hr>
<address>Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g Server at debian40server..local
Port 80</address>
</body></html>
```

Faites la même requête avec votre navigateur web, observez-vous une différence ?

```
http://xxx.xxx.xxx.xxx/tyuio
```

4.2.2 A vous de jouer !

Ecrivez les requêtes nécessaires pour observer les codes d'erreurs suivants. Pour gagner du temps, vous ne chercherez que pour les codes d'erreurs de la colonne de gauche, puis si vous avez encore du temps en fin de projet, vous viendrez faire ceux de droite. Attention, il ne suffit pas toujours de trouver une requête spécifique mais il faut que vous trouviez une requête pour une configuration serveur donnée. Vous indiquerez donc la configuration apache mise en place si celle-ci joue un rôle particulier.

- 200: OK,
- 403: Forbidden,
- 404: Not Found
- 400: Bad Request,
- 201: Created,
- 301: Redirection,
- 304: Not Modified,
- 401: Unauthorized.

5 Configuration avancée d'Apache http server (Obligatoire)

5.1 Virtual Host

Le principe des Serveurs Virtuels consiste à faire fonctionner un ou plusieurs serveurs Web (comme www.iut-nide1.com et www.iut-nice2.com) sur une même machine. Les serveurs virtuels peuvent être soit "par-IP" où une adresse IP est attribuée pour chaque serveur Web, soit "par-nom" où plusieurs noms de domaine se côtoient sur des mêmes adresses IP. L'utilisateur final ne perçoit pas qu'en fait il s'agit d'un même serveur physique.

Apache a été le précurseur des serveurs proposant cette méthode de serveurs virtuels basés sur les adresses IP. Ses versions 1.1 et suivantes ont toujours proposées ces deux méthodes de serveurs virtuels par-IP et par-nom. Cette deuxième méthode est parfois également appelée host-based ou serveur virtuel non-IP.

5.1.1 Serveurs virtuels par nom versus par IP

Les hébergements virtuels par IP utilisent l'adresse IP de la connexion afin de déterminer quel serveur virtuel doit répondre. Par conséquent, vous devez disposer d'adresses IP différentes pour chaque nom de domaine complet que vous hébergez. Avec un hébergement virtuel par nom, le serveur s'appuie sur les informations transmises par le client dans les en-têtes HTTP de ses requêtes. La technique présentée ici vous permet de disposer de serveurs virtuels différents partagés sur une même adresse IP.

L'hébergement virtuel par nom est habituellement plus simple, car il vous suffit de configurer votre serveur DNS pour que chaque domaine pointe sur l'adresse IP dont vous disposez, et de configurer votre serveur Apache HTTP afin qu'il reconnaisse ces domaines. Il réduit aussi la pénurie en adresses IP. Par conséquent, vous devriez utiliser l'hébergement virtuel par nom à moins d'avoir une raison spécifique de préférer l'hébergement virtuel par IP. Certaines de ces raisons vous sont exposées ci-après :

- Certains anciens navigateurs ne sont pas compatibles avec les serveurs virtuels par nom, car pour fonctionner, un client doit transmettre un champ d'en-tête HTTP Host. Cet en-tête est exigé pour HTTP/1.1, et peut être implémenté sur des navigateurs modernes HTTP/1.0 grâce à une extension.
- L'hébergement virtuel par nom ne peut pas être utilisé avec des serveurs sécurisés SSL à cause de la nature même du protocole SSL.
- Certains systèmes d'exploitation et équipements réseaux emploient des techniques de gestion de la bande passante qui ne peuvent pas différencier des domaines autrement que par des adresses IP séparées.

Dans notre cas, nous nous intéresserons, dans ce cours/TPs, uniquement aux serveurs virtuels par nom. Nous avons deux solutions pour mettre ceux-ci en place :

Solution 1 : modification du fichier *hosts* (nécessite les droits administrateurs)

Puisque nous n'avons pas de nom de domaine à affecter à notre serveur, nous allons devoir légèrement tricher pour mettre en œuvre plusieurs serveurs virtuels par nom et pouvoir observer le résultat.

Ajoutez les lignes suivantes dans le fichier *c:\windows\system32\drivers\etc\hosts* de Windows :

```
xxx.xxx.xxx.xxx www.iut-nice1.fr
xxx.xxx.xxx.xxx www.iut-nice2.fr
```

Bien sûr vous remplacerez les xxx.xxx.xxx.xxx par l'adresse ip de votre serveur http. Maintenant ouvrez la page http://www.iut-nice1.fr/ vous devriez tomber sur votre serveur http (de même avec http://www.iut-nice2.fr/). Le fichier *host* que nous venons de configurer sert à Windows de « DNS ». S'il y trouve le nom de domaine recherché,

il utilise l'adresse IP spécifiée en face, sinon il fait une requête DNS. Donc maintenant nous avons bien 2 noms de domaines utilisables.
Pensez bien à supprimer ces lignes dans le fichier *host* avant de quitter la salle, sans quoi ces noms de domaines seront bloqués.

Solution 2 : utiliser un service de DNS gratuit
Le but de cette manipulation est de créer deux noms de domaines (en fait on ne pourra créer que des noms de sous-domaines). Pour cela on utilisera le service DynDns (analogie de DNS Dynamique), qui permet aux particuliers et aux entreprises d'héberger leurs applications internet (web, ftp, mail, ldap, bases de données, ...) chez eux sans faire appel aux services des hébergeurs internet.
Rendez-vous sur http://www.dnsdynamic.org/ pour y créer deux « Free Domain Name » :

```
iut-nice1-prenom-nom.dnsd.me
iut-nice2-prenom-nom.dnsd.me
```

Il vous faudra vous créer un compte, puis modifier l'adresse IP associé au hostname en remplaçant celle existante par l'adresse IP de votre machine virtuelle. Une fois l'opération faite, patientez quelques minutes que le dns soit mis à jour puis ouvrez la page http://iut-nice1-prenom-nom.dnsd.me/ vous devriez tomber sur votre serveur http (de même avec http://iut-nice2-prenom-nom.dnsd.me/). Donc maintenant nous avons bien 2 noms de domaines utilisables.
Pensez bien à supprimer ces deux hostnames à la fin de ce cours, sans quoi ces noms de domaines seront bloqués.

Attention, dans la suite de ce cours/TP, on fait l'hypothèse que vous avez utilisez la solution 1. Si ce n'est pas le cas, adaptez l'énoncer avec les noms des domaines que vous avez créé.

5.1.2 Détail d'un fichier de configuration

Avant de poursuivre plus en avant la création de nos hôtes virtuels, regardons un peu ce que contient le fichier de configuration d'un hôte virtuel.
Ouvrez le fichier */etc/apache2/sites- available/default* et examinez l'ensemble des paramètres de ce fichier. Voici un exemple de fichier de configuration d'un hôte virtuel que vous pouvez obtenir. Pour les même raisons que précédemment, nous avons retiré l'ensemble des commentaires de ce fichier :

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

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

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
```

```
            Order allow,deny
            Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

La directive VirtualHost

<*VirtualHost*> et </*VirtualHost*> sont utilisés pour encapsuler un groupe de directives qui ne seront appliqués qu'à l'hôte virtuel qu'elles définissent. On spécifie l'adresse IP ou le nom de domaine auquel correspond l'hôte virtuel ainsi que le numéro de port. Dans l'exemple ci-dessus, (ainsi que par défaut), l'hôte virtuel est défini pour toutes les IPs (que le serveur http peut lire) et sur le port 80.

La directive ServerAdmin

ServerAdmin définie l'adresse de contact que le serveur inclut dans les messages d'erreurs qui seront retournés au client.

La directive DocumentRoot

Cette directive défini le répertoire d'origine à partir duquel httpd distribue les fichiers. À moins que l'url ne corresponde à un *Alias* (voir la directive *Alias*), le serveur ajoute le chemin de l'URL demandée à la racine du document pour faire le chemin au document. La racine (*DocumentRoot*) devrait être spécifiée sans un slash en fin de nom.

La directive Directory

<*Directory*> et </*Directory*> sont utilisés pour encapsuler un groupe de directives qui ne seront appliqués qu'au répertoire (et ces sous-répertoires) spécifié dans <*Directory nom_répertoire*>.

Combien y a-t-il de répertoires configurés dans le fichier donné en exemple ? Quels sont-ils ?

La directive AllowOverride

Quand le serveur trouve un fichier *.htaccess* (comme spécifié par *AccessFileName*) il doit connaître quelles directives déclarées dans ce fichier peuvent ignorer des directives de configuration précédentes. *AllowOverride* est valable seulement dans des sections <*Directory*> définies sans expressions régulières. Quand cette directive est mise à :

- ***None***, alors les fichiers *.htaccess* sont complètement ignorés. Dans ce cas, le serveur n'essayera pas même de lire des fichiers de *.htaccess* dans le système de fichiers.
- ***All***, le serveur prendra en compte toutes les directives dans le fichier *.htaccess*
- …

La directive Order

La directive *Order*, associée avec les directives *Allow* et *Deny*, pilote un système de contrôle d'accès à trois passages.

- Le premier passage traite toutes les directives *Allow* ou *Deny*, comme spécifié selon la directive *Order*.
- Le deuxième passage fait l'analyse syntaxique du reste des directives (*Allow* ou *Deny*).
- Le troisième passage s'applique à toutes les demandes qui ne correspondent à aucun des premiers deux.

Notez que toutes les directives *Allow* et *Deny* sont traitées, à la différence d'un pare-feu classique, où seulement la première directive correspondante est utilisée. De plus, l'ordre dans lequel les lignes apparaissent dans les fichiers de configuration n'est pas significatif - toutes les lignes *Allow* sont traitées comme un groupe et toutes les lignes *Deny* sont considérées comme un autre groupe.
L'ordre de traitement est un des deux suivants :

- **Allow,Deny :** En premier, toutes les directives *Allow* sont évaluées. Au moins une doit correspondre, ou la demande est rejetée. Ensuite, toutes les directives *Deny* sont évaluées. Si une d'elles correspond, la demande est rejetée. En fin, n'importe quelles demandes qui ne correspondent pas à un *Allow* ou une directive *Deny* sont refusées par défaut.
- **Deny,Allow :** En premier, toutes les directives *Deny* sont évaluées. Si une d'elles correspond, la demande est rejetée à moins qu'elle ne corresponde aussi à une directive *Allow*.

La directive Allow

La directive *Allow* défini quels hôtes peuvent avoir accès à un secteur du serveur. L'accès peut être contrôlé par hostname, l'adresse IP, la gamme d'adresse IP, ou selon d'autres caractéristiques de la requête du client. Le premier argument à cette directive est toujours *from*. Les arguments suivants peuvent prendre trois formes différentes.

- **Allow from all** : Pour permettre à tous les hôtes l'accès, soumis à la configuration des directives *Order* et *Deny*.
- **Allow from 'host'** : Pour permettre seulement aux hôtes particuliers ou les groupes d'hôtes d'avoir accès au serveur, l'hôte peut être spécifié dans n'importe lequel des formats suivants :
  - **Un nom de domaine partiel ou non** : *Allow from apache.org*
  - **Une adresse IP partielle ou non** : *Allow from 10.1.2.3* ou *Allow from 10.1*
  - **Un couple réseaux/netmask** : *Allow from 10.1.0.0/255.255.0.0*
  - **Un couple réseaux/nnn CIDR** (nnn= nombre de bit à 1) : *Allow from 10.1.0.0/16*
- **Allow from env=env-variable** : Pour permettre l'accès en fonction de l'existence ou de non de variable d'environnement.

La directive Deny

Cette directive permet de limiter l'accès au serveur d'être en fonction du *hostname*, de l'adresse IP, ou des variables d'environnement. Les arguments pour la directive *Deny* sont identiques à ceux de la directive *Allow*.

La directive Alias

La directive *Alias* permet aux documents d'être stockés dans le système de fichiers local ailleurs qu'à la racine définie par la directive *DocumentRoot*. Imaginons que nous souhaitions donner accès au répertoire */home/student/public_html/secret* en utilisant l'url http://www.iut-nice1.fr/secret/. Il suffirait pour cela d'ajouter les lignes suivantes dans le fichier *000-default (qui contiendrait la configuration d'un virtual host)*.

```
Alias /secret/ "/home/student/public_html/secret/"
    <Directory "/home/student/public_html/secret/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
       Order allow,deny
       Allow from all
    </Directory>
```

Créez le répertoire */home/student/public_html/secret* et ajoutez les lignes ci-dessus dans votre fichier *000-default*. Pensez à redémarrer votre serveur pour que votre nouvelle configuration soit prise en compte.

La directive ScriptAlias

La directive *ScriptAlias* a le même comportement que la directive *Alias*, sauf qu'en plus elle indique la liste des répertoires contenant les scripts CGI qui seront exécutés par le module *mod_cgi*.

Il est plus sûr d'éviter de placer des scripts CGI sous la racine *DocumentRoot*. Cela évitera de révéler accidentellement leur code source si jamais la configuration du serveur est changée.

La directive Options

La directive *Options* contrôle que les fonctions du serveur sont disponibles dans un répertoire particulier.

Options peut être mise à une des valeurs suivantes :

- ***None*** : dans ce cas on ne permet aucune fonction supplémentaire,
- ***All*** : toutes les options à part *Multivues*. C'est la valeur par défaut.
- ***ExecCGI*** : On permet l'exécution de scripts CGI utilisant *mod_cgi*.
- ***FollowSymLinks*** : le serveur suivra des l symboliques dans ce répertoire.
- ***Includes*** : autorise les inclusions du côté du serveur à l'aide du module mod_include.
- ***IncludesNOEXEC*** : les inclusions du côté du serveur, mais les *#exec cmd* et *#exec cgi* sont désactivés.
- ***Indexes*** : si une URL correspondant à un répertoire est demandée et qu'il ni à pas de *DirectoryIndex* (e.g., *index.html*) dans le répertoire, alors le module *mod_autoindex* pourra renvoyer une liste de répertoire.
- …

Normalement, si plusieurs directives *Options* sont appliquées à un répertoire, alors la plus spécifique est utilisée et les autres sont ignorés. Cependant si toutes les options de la directive *Options* sont précédées par le symbole + ou - les options sont fusionnées. N'importe quelle option précédée par un + est ajoutée aux options actuellement actives et n'importe quelle option précédée par un - est enlevée des options actuellement actives.

- **Attention**, le mélange d'options avec un + ou - avec des sans symbole n'est pas une syntaxe valable. Il est probable que vous obteniez des résultats inattendus.

Par exemple, sans aucun + et - symboles où seul l'option *Includes* sera active pour le répertoire */web/docs/spec*.

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
   Options Includes
</Directory>
```

Cependant si la deuxième directive *Options* utilise des symboles + et - alors les options *FollowSymLinks* et *Includes* seront actives pour le répertoire */web/docs/spec*.

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
   Options +Includes -Indexes
</Directory>
```

Revenons maintenant à notre problème premier, comment créer des serveurs virtuels avec une même IP.

5.1.3 Utilisation de serveurs virtuels par nom

Pour utiliser des serveurs virtuels par nom, vous devez désigner l'adresse IP (et si possible le port) sur le serveur devant accepter les requêtes pour des domaines. Cette configuration utilise la directive *NameVirtualHost*. Dans un cas normal, où n'importe quelle adresse IP peut être utilisée, vous pouvez ajouter * comme argument de la

directive *NameVirtualHost*. Si vous prévoyez d'utiliser de multiples ports (comme l'emploi de SSL), vous devriez ajouter le port à cet argument tel que *:80. Si la directive suivante n'est pas dans votre fichier *ports.conf* ajoutez là.

```
NameVirtualHost *:80
```

L'étape suivante est la création d'une section <*VirtualHost*> pour chacun des serveurs à créer. L'argument de la directive <*VirtualHost*> doit être le même que celui de la directive *NameVirtualHost* (c'est-à-dire l'adresse IP ou * pour toutes les adresses). Dans chaque section <*VirtualHost*>, vous devez définir au minimum une directive *ServerName* pour désigner le serveur concerné et une directive *DocumentRoot* pour préciser l'emplacement sur le système de fichiers du contenu de ce serveur. Créez deux copies du fichier *default* dans le répertoire */etc/apache2/sites-available/* que vous nommerez *nice1* et *nice2*. Ajoutez la ligne suivante dans le fichier *nice1* entre la balise <*VirtualHost *:80*> et la directive *DocumentRoot*

```
ServerName www.iut-nice1.fr
```

Modifiez la directive *DocumentRoot* comme indiqué par la ligne ci-dessous :

```
DocumentRoot /var/www/html/nice1
```

Pensez à créer le répertoire */var/www/html/nice1* et ajoutez-y un fichier index.html qui affiche « Nice 1 ».
On recommence pour *nice2*. Ajoutez la ligne suivante dans le fichier *nice2*entre la balise <*VirtualHost *:80*> et la directive *DocumentRoot*

```
ServerName www.iut-nice2.fr
```

Modifiez la directive *DocumentRoot* comme indiqué par la ligne ci-dessous :

```
DocumentRoot /var/www/html/nice2
```

Pensez à créer le répertoire */var/www/html/nice2* et ajoutez-y un fichier index.html qui affiche « Nice 2 »

Le serveur « principal » disparaît

Dans les configurations actuelles, on n'utilise plus le serveur « principal », mais uniquement des VirtualHost. Mais cela pourrait être le cas sur une plus ancienne version d'apache httpd.

Originellement, on n'utilisait pas de VirtualHost ; on ne définissait qu'un seul serveur, directement dans le fichier httpd.conf (ou son équivalent). Donc quand vous ajoutiez des serveurs virtuels à un serveur Web existant, vous deviez également créer une section <*VirtualHost*> redéfinissant ce serveur existant (car celui-ci était désactivé). Les directives *ServerName* et *DocumentRoot* incluses dans ce serveur virtuel devaient être les mêmes que pour les directives globales *ServerName* et *DocumentRoot*. Et ce serveur virtuel était positionné en premier dans le fichier de configuration pour en faire le serveur par défaut.

Serveurs virtuels par IP

Vous pouvez également spécifier une adresse IP explicite à la place de * dans les deux directives *NameVirtualHost* et <*VirtualHost*>. Par exemple, cette méthode est utile si vous souhaitez faire tourner quelques serveurs virtuels par nom sur une même adresse IP, et d'autres, soit par IP, soit basés sur un autre jeu de serveurs virtuels par nom sur une autre adresse IP.

Plusieurs noms pour un même serveur virtuel

Plusieurs serveurs sont accessibles par plus d'un nom. Il suffit de placer la directive *ServerAlias* dans une section <*VirtualHost*>. Par exemple, dans la section <*VirtualHost*> de votre fichier *nice1*, ajoutez la directive *ServerAlias* comme ci-dessous. Cela indique aux utilisateurs les autres noms permis pour accéder au même site Web :

```
ServerAlias iut-nice1.fr *.iut-nice1.fr
```

Ainsi, toutes les requêtes portant sur un domaine *iut-nice1.fr* seront servies par le serveur virtuel *www. iut-nice1.fr*. Les caractères joker * et ? peuvent être utilisés pour les correspondances. Bien entendu, vous ne pouvez pas inventer des noms et les placer dans une directive *ServerName* ou *ServerAlias*. Tout d'abord, votre serveur DNS doit

être correctement configuré pour lier ces noms à une adresse IP associée avec votre serveur. Dans notre cas, pour pouvoir tester ces noms, il va falloir remodifier le fichier *c:\windows\system32\drivers\etc\hosts*.

Maintenant, lorsqu'une requête arrive, le serveur va d'abord tester si elle utilise une adresse IP qui correspond à *NameVirtualHost*. Si c'est le cas, il regardera chaque section <*VirtualHost*> avec l'adresse correspondante et essaiera d'en trouver une où le nom de domaine requis correspond à *ServerName* ou *ServerAlias*. S'il en trouve une, il utilisera sa configuration pour le serveur. Si aucun serveur virtuel ne correspond, alors le premier serveur virtuel listé dont l'adresse IP correspond sera employé. En conséquence, le premier serveur virtuel listé est le serveur virtuel default. La directive *DocumentRoot* du serveur principal ne sera jamais employée lorsqu'une adresse IP correspond dans une directive *NameVirtualHost*. Si vous ne voulez pas avoir de configuration spéciale pour les requêtes qui ne sont pas attachées à un serveur virtuel en particulier, mettez cette configuration dans une section <*VirtualHost*> que vous placerez en premier dans le fichier de configuration.

Activez vos 2 nouveaux sites à l'aide de la commande *a2ensite* qui est similaire à celle utilisé pour les modules (*a2dissite* pour désactiver un virtualhost).

Testez les 3 adresses suivantes et vérifiez qu'à chaque fois vous êtes bien sur un serveur virtuel différent :

- http://www.iut-nice1.fr/
- http://www.iut-nice2.fr/
- http://xxx.xxx.xxx.xxx/ (avec l'IP de votre serveur)

Sans testez l'action sur votre serveur, dites ce qui arrive si on désactive le site par *default* (c'est-à-dire si on efface le fichier */etc/apache2/sites-enabled/000-default)* et qu'on ouvre l'url suivante sur un navigateur. Pourquoi obtient-on ceci ?

- http://xxx.xxx.xxx.xxx/ (avec l'IP de votre serveur)

On notera que pour désactiver un site, il est préférable d'utiliser la commande *a2dissite*.

5.2 Vérification des connaissances (Obligatoire)

Si vous avez bien compris les TPs, vous devriez être capable de répondre facilement aux questions suivantes. Si ce n'est pas le cas, il est conseillé de retravailler la section du TP qui correspond aux questions dont vous n'êtes pas sûres de réponse.

1. Quelle directive spécifie le port TCP sur lequel écouter ? Quel est le port par défaut ?
2. Il y a-t-il d'autres fichiers de configuration chargés depuis apache2.conf ? Où sont-ils placés ? Donnez en la liste.
3. Sous quelle identité unix (utilisateur et groupe) le serveur va-t-il s'exécuter ? Donner les UID et GID correspondants.
4. Quel est le répertoire racine pour les documents (pages) servis ?
5. Quelle page Apache2 renvoie-t-il lorsque l'URL demandée correspond à un répertoire ?
6. Comment sont traitées les URL de la forme http://serveur/cgi-bin/toto ?
7. Quels sont les fichiers de logs générés ? Où sont-ils placés ? Quel est leur format et comment est-il contrôlé ?

5.3 Les fichiers .htaccess : authentification et bien plus (Conseillé)

Les fichiers *.htaccess* sont 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.

5.3.1 Intérêt des fichiers .htaccess

Les fichiers *.htaccess* peuvent être utilisés dans n'importe quel répertoire virtuel ou sous-répertoire. Les principales raisons d'utilisation des fichiers *.htaccess* sont :

- Gérer l'accès à certains fichiers.
- Ajouter un mime-type.
- Protéger l'accès à un répertoire par un mot de passe.
- Protéger l'accès à un fichier par un mot de passe.
- Définir des pages d'erreurs personnalisées.
- Principe des fichiers *.htaccess*

Le fichier *.htaccess* est placé dans le répertoire dans lequel il doit agir. Il agit ainsi sur les permissions du répertoire qui le contient et de tous ses sous-répertoires. Vous pouvez placer un autre fichier *.htaccess* dans un sous-répertoire d'un répertoire déjà contrôlé par un fichier *.htaccess*. Le fichier *.htaccess* du répertoire parent reste en « activité » tant que les fonctionnalités n'ont pas été réécrites. Pensez à bien positionner la directive *AllowOverride* pour que vos fichiers *.htaccess* soient pris en compte.

5.3.2 Empêcher l'accès à des ressources

Un fichier *.htaccess* est composé de deux sections :

1. Une première section contient les chemins vers les fichiers contenant les définitions de groupes et d'utilisateurs :

```
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /repertoire/de/votre/fichier/.FichierDeGroupe
AuthName "Accès protégé"
AuthType Basic
```

- ***AuthUserFile*** définit le chemin d'accès absolu vers le fichier de mot de passe.
- ***AuthGroupFile*** définit le chemin d'accès absolu vers le fichier de groupe.
- ***AuthName*** entraîne l'affichage dans le navigateur Internet de : « Tapez votre nom d'utilisateur et votre mot de passe. Domaine: "Accès protégé" »
- ***AuthType*** Basic précise qu'il faut utiliser *AuthUserFile* pour l'authentification.

2. Une seconde section contient la définition des conditions d'accès :

```
<Limit GET POST>
Require valid-user {instruction d'accès à satisfaire}
</Limit>
```

- **La balise *LIMIT*** possède en attribut la valeur *GET* (en majuscule) et/ou la valeur *POST*, afin de définir le type de méthode du protocole *HTTP* auxquelles la restriction s'applique
- ***Require valid-user*** précise que l'on autorise uniquement les personnes identifiées. Il est également possible de préciser explicitement le nom des personnes autorisées à s'identifier : *require user {username}*

Le chemin d'accès vers les fichiers de mots de passe et de groupes est de la forme suivante :

```
/repertoire1/repertoire2/.../.FichierDeMotDePasse
```

5.3.3 Protéger un répertoire par un mot de passe

Il s'agit d'une des applications les plus utiles du fichier *.htaccess* car elle permet de définir de façon sûre (à l'aide d'un login et d'un mot de passe) les droits d'accès à des fichiers par certains utilisateurs. La syntaxe est la suivante :

```
AuthUserFile {emplacement du fichier de mot de passe}
AuthGroupFile {emplacement du fichier de groupe}
AuthName "Accès protégé"
AuthType Basic
<LIMIT GET POST>
Require valid-user
</LIMIT>
```

- La directive *AuthUserFile* permet de définir l'emplacement du fichier contenant les logins et les mots de passe des utilisateurs autorisés à accéder à une ressource donnée.
- La directive *AuthGroupFile* permet de définir l'emplacement du fichier contenant les groupes d'utilisateurs autorisés à s'identifier. Il est possible d'outrepasser cette déclaration en déclarant le fichier suivant : */dev/null*.

Voici un exemple de fichier *.htaccess* :

```
ErrorDocument 403 http://www.iut-nice1.fr/error403.html
AuthUserFile /var/conf/www/.pass
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site Nice 1"
AuthType Basic
<LIMIT GET POST>
Require valid-user
</LIMIT>
```

Créez un fichier *.htaccess* qui demande de s'identifier pour accéder au répertoire secret que vous avez créé. Utilisez un fichier de mot de passes contenant au moins 4 noms et vérifiez que seulement ceux-ci fonctionnent.

Le fichier de mot de passe est un fichier texte devant contenir sur chacune de ses lignes le nom de chaque utilisateur suivi des deux points (:), puis du mot de passe crypté (solution recommandée) ou en clair. Ce fichier de mot de passe ne devrait pas être mis dans un répertoire virtuel Internet.

Voici un exemple de fichier de mots de passe non chiffrés (ici *.pass*). ***Ne pas utiliser ce fichier mais utiliser la version chiffrée*.** Les versions non chiffrées ne sont plus supportées par défaut par les navigateurs actuels.

```
utilisateur:user
gaetan:passpass
```

Voici le même fichier contenant des mots de passe chiffrés :

```
utilisateur:YYam8h/9OU2aQ
gaetan:J2Jt7wQJ0piNA
```

5.3.4 Crypter les mots de passe

Apache fournit un outil permettant de générer facilement des mots de passe cryptés (aussi bien sous Windows que sous Unix), il s'agit de l'utilitaire *htpasswd* accessible dans le sous-répertoire *bin* d'Apache (pour nous */usr/bin htpasswd*). La syntaxe de cet utilitaire est la suivante :

- Pour créer un nouveau fichier de mots de passe :

```
htpasswd -c {chemin du fichier de mot de passe} utilisateur
```

- Pour ajouter un nouvel utilisateur/mot de passe à un fichier existant :

```
htpasswd {chemin du fichier de mot de passe} utilisateur
```

5.3.5 Empêcher l'accès à un répertoire par un domaine

La syntaxe pour bloquer l'accès d'un répertoire par un domaine est la suivante :

```
Order (allow,deny ou deny,allow)
Allow (all, [liste de domaine])
Deny (all, [liste de domaine])
```

Exemple :

```
Order deny,allow
Deny from .domaine.com
```

Toutes les requêtes provenant du domaine *.domaine.com* ne pourront avoir accès aux ressources comprises dans le répertoire et ses sous-répertoires. La commande *Order* sert à préciser explicitement que la commande *Deny* va bien annuler l'effet de *Allow* et non l'inverse.

Voici un exemple de restriction d'accès plus complet:

```
ErrorDocument 403 http://www.iut-nice1/accesrefuse.html
AuthUserFile /var/conf/www/.pass
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site Nice 1"
AuthType Basic
<LIMIT GET POST>
order deny,allow
deny from all
allow from 123.148.12.2
require user utilisateur gaetan
</LIMIT>
```

Dans ce cas, l'accès ne sera possible que pour les utilisateurs « utilisateur » et « gaetan » à partir de l'adresse IP 123.148.12.2 et avec le bon mot de passe. Créez un fichier *.htaccess* qui demande de s'identifier pour accéder à l'url http://www.iut-nice1.fr/ et qui limite cet accès à l'utilisateur thomas.

5.3.6 Empêcher l'accès à fichier particulier

Par défaut, Apache applique les restrictions du fichier *.htaccess* à l'ensemble des fichiers du répertoire dans lequel il se trouve ainsi qu'à tous les fichiers contenus dans ses sous-répertoires. Il est également possible de restreindre l'accès pour un ou plusieurs fichiers du répertoire grâce à la balise *<Files>*.

Voici un exemple de restriction aux fichiers *secret.html* et *secret2.html* :

```
<Files secret.html>
AuthUserFile /var/conf/www/.pass
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site Nice 1"
AuthType Basic
<LIMIT GET POST>
require user gaetan
</LIMIT>
</Files>
```

```
<Files secret2.html>
AuthUserFile /var/conf/www/.pass
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site Nice 1"
```

```
AuthType Basic
<LIMIT GET POST>
require valid-user
</LIMIT>
</Files>
```

Créez un fichier *.htaccess* qui demande de s'identifier pour accéder à l'url http://www.iut-nice1.fr/secret/info.htm et qui limite cet accès à l'utilisateur thomas.

Il ne faut utiliser qu'une seule balise *<Files>* par fichier. Sinon, l'erreur suivante est reportée dans le fichier de log des erreurs :

```
.htaccess: Multiple arguments not (yet) supported.
```

Pour info, depuis Apache 1.3, il est conseillé d'utiliser la balise *<FilesMatch>* à la place de la balise *<Files>*. Cette nouvelle balise ne supporte aussi qu'un seul argument mais on peut traiter plusieurs fichiers grâce à une expression régulière.

5.3.7 Empêcher l'accès à un type de fichiers par un domaine

```
<Files *.png>
Deny from .domaine.com
</Files>
```

Toutes les requêtes provenant du domaine *.domaine.com* ne pourront avoir accès aux images, dont l'extension est .png, comprises dans le répertoire et ses sous-répertoires. Créez un fichier *.htaccess* qui bloque la lecture des fichiers *.gif et *.txt dans le répertoire secret pour tous les utilisateurs. Vous pouvez copiez un fichier depuis internet sur votre machine virtuelle en utilisant la commande suivante :

```
> wget http://portail.unice.fr/jahia/txt/inc/img/logo_unice.gif
```

5.3.8 Autoriser l'accès à un groupe de fichiers par un domaine et un pays

```
<Files php*>
Order deny,allow
Deny from all
Allow from .iut-nice1.fr
Allow from .org
</Files>
```

Toutes les requêtes provenant du domaine *.iut-nice1.fr* ou des domaines ayant la terminaison *.org* pourront avoir accès aux fichiers commençant par php (par exemple, les fichiers *phpindex.html*, *phplogo.gif*) compris dans le répertoire et ses sous-répertoires.

5.3.9 Protéger un répertoire par un login

Cette méthode (beaucoup moins sûre que la précédente) permet une authentification de bas niveau uniquement par le nom de l'utilisateur. La syntaxe est la suivante :

```
Require (user [liste des utilisateurs], group [liste des groupes], valid-user)
```

Voici un exemple de ligne du fichier .htaccess :

```
Require user gaetan student thomas paul
```

Attention ici user est un mot clef et non le nom d'un utilisateur !!!

Tout utilisateur souhaitant rentrer dans le répertoire ou un de ses sous-répertoires sera refusé sauf s'il s'identifie en donnant un nom figurant dans la liste.

5.3.10 Obliger un utilisateur à satisfaire à au moins une des conditions

Voici la syntaxe :

```
Satisfy (any, all)
```

```
Order Allow, Deny
Deny from all
Allow from .free.fr
Require user gaetan student thomas paul
Satisfy Any
```

Ce qui signifie que l'accès au répertoire sera bloqué pour tout le monde à l'exception des personnes qui s'identifient et des requêtes provenant du domaine .free.fr.

5.3.11 Ajouter un Mime-Type à un répertoire

Un type MIME (en anglais MIME type) est un ensemble de types de fichiers standard, permettant d'associer une extension de fichier donnée à une application, afin d'automatiser le lancement de l'application.

La syntaxe est la suivante :

```
AddType (mime/type [liste d'extension])
Voici un exemple de mise en oeuvre du fichier .htaccess :
AddType image/x-photoshop PSD
AddType application/x-httpd-php .php3
AddType application/x-httpd-php .htm
```

Le serveur enverra au navigateur Internet le fichier en lui disant de lancer le programme *Photoshop* (s'il est installé sur votre machine) et de lui donner le fichier. Habituellement, ceci est utilisé pour des fichiers nécessitant un Plug-in particulier non reconnu par votre navigateur.

Cette commande permet aussi d'annuler tout élément prédéfini. Ainsi, vous pouvez enregistrer un fichier *.wav* avec une extension *.gif* et préciser au navigateur de considérer les fichiers *.gif* comme des fichiers audio ! En pratique, on pourra donc utiliser cette commande pour ordonner à l'interpréteur *PHP* d'évaluer d'autres extensions de fichier, *.htm* par exemple.

5.3.12 Forcer tous les fichiers d'un répertoire à un Mime-Type

Voici la syntaxe à adopter :

```
ForceType (mime/type)
```

Par exemple avec la ligne suivante, tous les fichiers du répertoire contenant le fichier *.htaccess* seront considérés comme étant des fichiers *.jpg* quelle que soit leur extension :

```
ForceType image/jpg
```

Ce type de commande ne peut être utilisé dans les bornes !

5.3.13 Définir les extensions de fichiers par défaut

La syntaxe à suivre est :

```
DefaultType (mime/type)
```

Par exemple

```
DefaultType text/html
```

Cette option vous permet de définir le comportement par défaut du navigateur face à des extensions lui étant inconnues. Ici, il prendra tout fichier inconnu en tant que document *HTML*.

5.3.14 Personnalisation des messages d'erreurs

Il s'agit d'une fonctionnalité pratique car elle permet de définir une page par défaut pour un type d'erreur donné. Cela permet d'une part de guider l'utilisateur au lieu d'afficher la banale page d'erreur du navigateur, ainsi que d'égayer la navigation même en cas d'erreur.

```
ErrorDocument (code-à-3-chiffres [nom du fichier ou texte ou url])
```

Les deux lignes suivantes permettent de définir des pages d'erreurs personnalisées au cas où l'accès à un document serait interdit ou bien que le document n'existe pas :

```
ErrorDocument 403 /erreurs/403.php3
ErrorDocument 404 /erreurs/404.php3
```

Ceci vous permet de donner un message d'erreur personnalisé remplaçant les fichiers fournis avec le navigateur. Voici quelques-unes des erreurs les plus courantes à personnaliser :

```
401 Unauthorized : la personne n'a pas passé avec succès l'identification.
403 Forbidden : le serveur n'a pas le droit de répondre à votre requête.
404 Not Found : le serveur n'a pas trouvé le document souhaité.
```

5.3.15 Changer le fichier index par défaut

Le fichier *index.html* est le fichier qui est affiché lorsqu'aucun nom de fichier n'est défini dans l'URL. Cela permet d'éviter que le navigateur liste l'ensemble des fichiers contenus dans le répertoire (pour des raisons de confidentialité). On a déjà vu que la syntaxe pour effectuer ce type d'opération est la suivante :

```
DirectoryIndex (fichiers)
```

Voici un exemple de mise en application :

```
DirectoryIndex index.php index.html index.phtml /erreurs/403.php
```

Lorsque vous essayez d'accéder au répertoire sans préciser la page à afficher, Apache va avoir recours à la directive DirectoryIndex. En général, par défaut, cette directive pointe vers *index.html* puis *index.htm*. Dans l'exemple ci-dessus, Apache va commencer par chercher *index.php*, puis *index.html*, et ensuite *index.phtml*. Si aucun de ces trois fichiers existent, la page *403.php* (se trouvant dans la racine) sera affichée pour éviter de lister le répertoire.

5.4 Sécurité et SSL (conseillé)

L'objectif de cette partie et d'activer le support des pages commençant par *https://*. Pour accepter les requêtes SSL, Apache a besoin de deux fichiers : une clé pour le serveur *cle_ssl.key* et un certificat signé *crt_ssl.crt*. Les noms des fichiers n'ont pas d'importance. La signature de ce certificat est réalisée par un organisme de certification tiers (tel que *Verisign* ou *Thawte*). Cependant vous pouvez signez-vous même votre certificat, la seule différence sera un avertissement par le navigateur lors de l'accès à une ressource SSL de votre serveur. La sécurité est la même, que le certificat soit signé par vous-même ou pas un organisme.

5.4.1 Installation du module ssl pour apache

Nous allons, dans un premier temps, installer les modules nécessaire pour faire fonctionner *Apache* en sécurisé. Tout d'abords nous avons besoin du module *OpenSSL* (c'est une boite à outils de chiffrement comportant deux bibliothèques (une de cryptographie générale et une implémentant le protocole SSL), ainsi qu'une commande en ligne). Normalement le module doit déjà être présent. Si ce n'est pas le cas installez-le avec la commande :

```
> apt-get install openssl
ou
> aptitude install openssl
```

Maintenant nous allons devoir installer le module *ssl* pour *Apache*. Normalement le module *ssl* (c'est-à-dire les fichiers *ssl.conf* et *ssl.load*) se trouve déjà dans */etc/apache2/mods-available/*. Activez le module ssl. Notez que vous pouvez aussi activer ou désactiver les modules Apaches à l'aide des commandes suivantes :

```
> a2enmod {nom_du_module}
> a2dismod {nom_du_module}
```

Nous supposerons dans la suite que le fichier *openssl.cnf* est dans */etc/ssl/* (ce qui devrait être normalement le cas). Vérifiez que c'est bien le cas pour vous.

5.4.2 Régler le serveur pour qu'il écoute (aussi) sur le port 443

Par défaut, Apache2 est configuré pour écouter sur le port 80. Vous pouvez vérifier les ports actifs sur votre machine à l'aide de la commande :

```
netstat -nlt
```

Or le protocole SSL a besoin d'émettre sur un port spécifique pour pouvoir fonctionner, celui qui est adopté en général est le port 443. Pour cela ajoutez la directive suivante dans le fichier */etc/apache2/ports.conf (sauf si celle-ci est déjà présente)* :

```
Listen 443
```

Redémarrez *Apache* et vérifiez si le serveur écoute bien sur le port 443.

5.4.3 Création des clés et certificats avec openssl

Pour créer votre clé et votre certificat, tapez commandes les suivantes dans un terminal :

```
> mkdir /tmp/ssl_conf
> cd /tmp/ssl_conf
> openssl req -config /etc/ssl/openssl.cnf -new -out csr_ssl.csr
```

Là il vous demande un *passphrase*, entrez un mot de passe dont vous vous souviendrez. Finissez, en entrant des informations régionales (exemple : FR/PACA/Nice/Unice/LP/Gaetan Rey/Gaetan.rey@unice.fr/...). Ensuite, tapez :

```
> openssl rsa -in privkey.pem -out cle_ssl.key
> openssl x509 -in csr_ssl.csr -out crt_ssl.crt -req -signkey cle_ssl.key -days 3650
> openssl x509 -in crt_ssl.crt -out crt_ssl.der.crt -outform DER
```

Notez que mon certificat est valable 3650 jours (presque 10 ans)

Copiez la clé et le certificat dans les dossiers ssl d'Apache (**vous devrez peut-être les créer**) :

```
> cd /tmp/ssl_conf
> cp crt_ssl.crt /etc/apache2/ssl.crt/
> cp cle_ssl.key /etc/apache2/ssl.key/
```

5.4.4 Création du fichier de configuration

Créez un nouveau *virtualhost* :

```
<VirtualHost *:443>
      DocumentRoot /var/www/html/site_ssl
      ServerName www.iut-nice1.fr
   <Directory /var/www/html/site_ssl/>
       Options Indexes MultiViews FollowSymLinks
       AllowOverride None
       Order deny,allow
       Deny from all
       Allow from all
   </Directory>
      SSLEngine on
      SSLCertificateFile /etc/apache2/ssl.crt/crt_ssl.crt
     SSLCertificateKeyFile /etc/apache2/ssl.key/cle_ssl.key
</VirtualHost>
```

6 Installation et configuration d'un forum phpbb (Conseillé)

Comme le montre le panorama des forums PHP gratuits du site http://www.journaldunet.com/ (février 2004), phpBB est aujourd'hui « le leader » des forums PHP. C'est pourquoi nous avons décidé de vous présenter en détail les différentes étapes de la mise en route de phpBB. Vous pourrez trouver sur Wikipédia un comparatif plus poussé des différents logiciels de gestion de forum en php, asp ou autre.

6.1 Les pré-requis

il faut disposer d'un serveur http acceptant les scripts PHP et donnant accès à l'une des bases reconnues par le script du forum phpBB, à savoir MySQL, PostgreSQL, MS SQL Server ou MS Access... Nous installerons ce forum dans notre machine virtuelle linux-debian. Commençons par ajouter mysql à notre installation :

```
> apt-get install mysql-server
ou
> aptitude install mysql-server
```

Nous devons également installer les modules php pour apache. Pour cela, vous devez installer un interpréteur php à l'aide de la commande suivante :

```
> apt-get install php5
ou
> aptitude install php5
```

Ajoutez la ligne suivante pour que les fichiers php soient bien interprétés par apache.

```
AddType application/x-httpd-php .php .php5
```

Nous allons également installer le package suivant pour faire le lien entre php5 et mysql

```
> apt-get install php5-mysql
ou
> aptitude install php5-mysql
```

Connectez-vous avec les droits root sous mysql. Créez une nouvelle base nommée *forum* puis créez un nouvel utilisateur ayant les droits sur cette base.

```
> mysql
mysql> CREATE DATABASE forum;
mysql> USE forum;
mysql> GRANT ALL PRIVILEGES ON forum.* TO 'phpuser'@'localhost' IDENTIFIED BY 'passforum'
WITH GRANT OPTION;
mysql> COMMIT;
mysql> QUIT;
```

6.2 Récupération et mise en ligne

Le script correspondant au forum phpBB se trouve sur le site officiel, vous pouvez le télécharger avec la commande suivante :

```
Directement depuis le serveur (attention problème de téléchargement possible avec wget) :
> wget https://www.phpbb.com/files/release/phpBB-3.0.12.tar.bz2
ou
Depuis votre machine, téléchargez le même fichier puis déposé le sur le serveur en utilisant
WinSCP (ou tout autre logiciel équivalent)
```

Une fois le script téléchargé, il faut le décompresser dans un répertoire. Cependant l'archive étant en bz2, nous devons d'abord installer le logiciel bzip2 à l'aide de la commande suivante. Installez bzip2 puis décompresser votre fichier dans un répertoire.

```
> aptitude install bzip2
> tar –xjvf phpBB-3.0.12.tar.bz2
```

Un répertoire du type *phpBB3* est créé automatiquement.
Nous allons maintenant récupérer le pack de langue français. Téléchargez-le puis décompressez le fichier *lang_french.tar.gz* dans le dossier *phpBB3/language* qui existe déjà. Vous devez après décompression avoir deux dossiers dans ce répertoire *language* : *fr* et *en*.

```
Directement depuis le serveur :
> wget https://www.phpbb.com/customise/db/download/id_91611
ou
Depuis votre machine, téléchargez le même fichier puis déposé le sur le serveur en utilisant
WinSCP (ou tout autre logiciel équivalent)
puis
> tar –xvf lang_fr.tar.gz
```

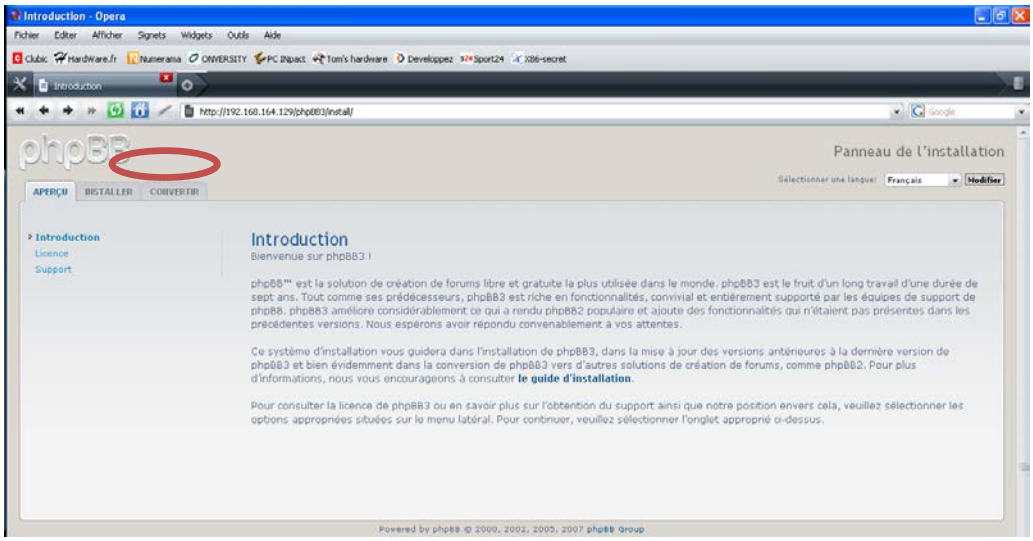
Vous pouvez également faire la même chose pour les thèmes subsilver2 et proSilver disponible aux adresses suivantes :

- http://www.phpbb.com/files/language_packs_3ox/subsilver2_fr.tar.gz
- http://www.phpbb.com/files/language_packs_3ox/prosilver_fr.tar.gz

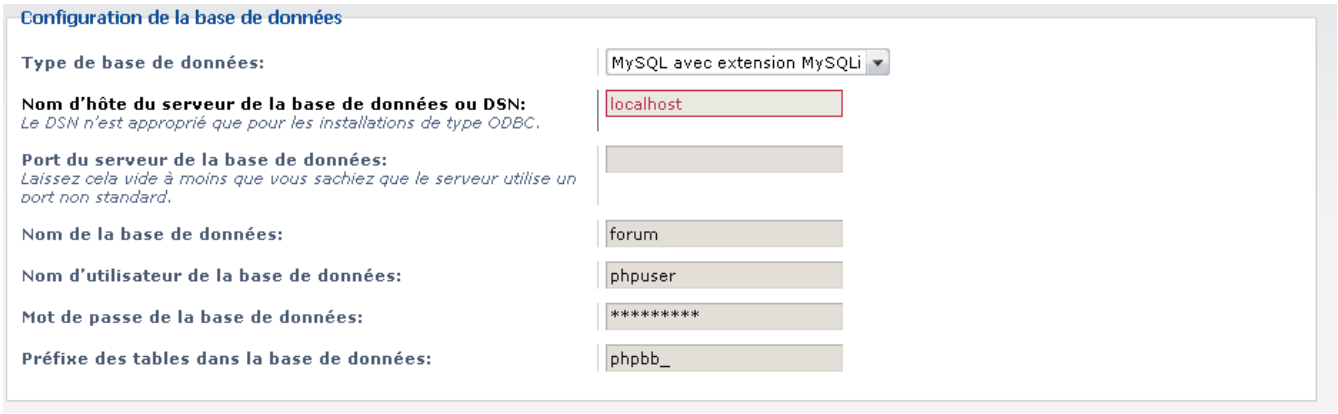
6.3 Configuration

Avant toute autre chose, vous allez créer un alias dans un des hosts virtuels de manière à ce qu'on puisse accéder au forum (le répertoire phpBB3) à l'aide d'une URL du type http://xxx.xxx.xxx.xxx/forum/

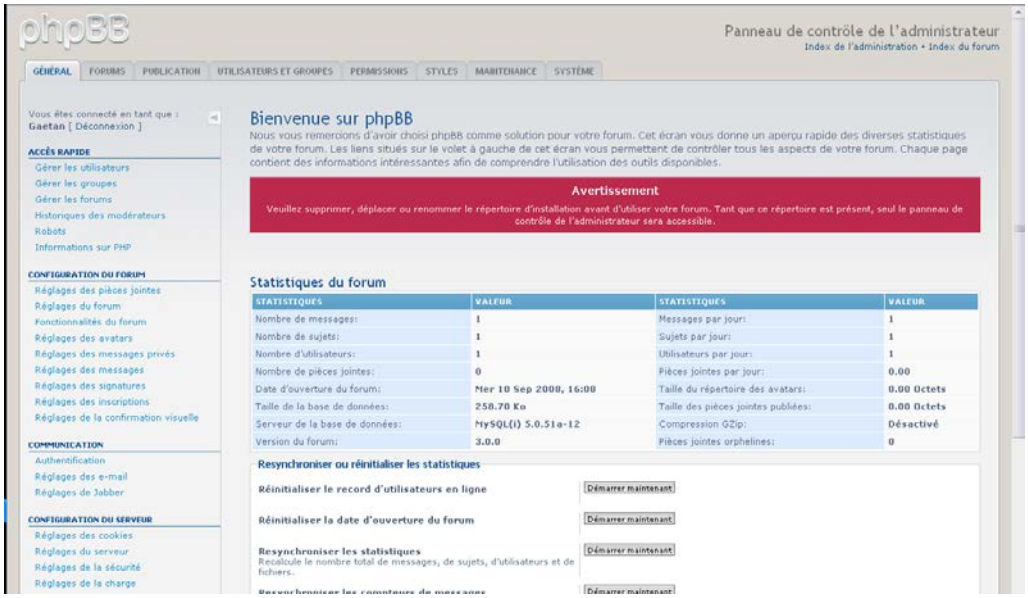
Maintenant allez sur la page http://xxx.xxx.xxx.xxx/forum/install/ depuis votre navigateur. Allez sur l'onglet *INSTALLER* et suivez les instructions.



Entrez les données suivantes pour la configuration de votre base de données :



Finissez tranquillement de remplir les différents champs (vous désactiverez les fonctions SMTP pour ne pas avoir de problème). Une fois arrivé à la page suivante, vous avez terminé l'installation. Bravo ;). Il faut maintenant « *supprimer, déplacer ou renommer le répertoire d'installation avant d'utiliser votre forum. Tant que ce répertoire est présent, seul le panneau de contrôle de l'administrateur sera accessible.* »
Rendez-vous sur la page http://xxx.xxx.xxx.xxx/forum/ pour vérifier si votre forum est bien fonctionnel.



Modifier la configuration de votre serveur pour que votre forum soit accessible via l'adresse http://forum.iut-nice1.fr.

7 Installation et configuration d'un wiki (Optionnel)

Un wiki est un logiciel de la famille des systèmes de gestion de contenu de site web rendant les pages web modifiables par tous les visiteurs y étant autorisés. Il facilite l'écriture collaborative de documents avec un minimum de contraintes. Créé en 2001, Wikipédia est devenu peu à peu le plus visité des sites web écrits avec un wiki.
Le mot « wiki » vient du redoublement hawaiien wiki wiki, qui signifie « rapide ». Les wikis ont été inventés en 1995 par Ward Cunningham, pour réaliser la section d'un site sur la programmation informatique, et qu'il a appelé WikiWikiWeb. Au milieu des années 2000, les wikis ont atteint un bon niveau de maturité. Ils sont depuis lors associés à ce qui est dénommé Web 2.0. [Wikipédia]

7.1 A vous de jouer

Parmi les différents Wiki décrit ci-dessous, choisissez celui que vous souhaitez et installez-le de manière à ce qu'il soit accessible via l'adresse suivante http://wiki.iut-nice1.fr/.

- DokuWiki est un Wiki, conforme aux standards, simple à utiliser, dont le but est de créer des documentations de toute sorte. Il est destiné aux équipes de développement, aux travaux de groupe et aux petites entreprises. Il possède une syntaxe simple mais puissante qui assure la lisibilité des fichiers de données en dehors du Wiki, et facilite la création de textes structurés. Toutes les données sont stockées dans des fichiers textes – aucune base de donnée n'est requise.

- MediaWiki est un logiciel libre développé à l'origine pour Wikipédia et utilisé aujourd'hui par de nombreux autres projets de l'association à but non lucratif Wikimedia Foundation ainsi que par d'autres sites reposant sur la technologie wiki, sous license GNU General Public License (GPL). Il est utilisé par Wikipédia et d'autres projets de la fondation Wikimédia, ainsi que par bien d'autres sites et wikis.

- PmWiki est un système de type wiki pour la création et l'entretien collectif de sites Internet. Les pages PmWiki ont le même aspect et fonctionnent comme des pages Internet ordinaires, sauf qu'elles possèdent un lien pour "Éditer" ce qui permet de modifier ou d'ajouter facilement des pages à un site, en utilisant les règles d'édition de base. Vous n'avez pas besoin de connaître le langage HTML ou les CSS. L'édition des pages peut être laissée ouverte à tout public ou restreinte à un petit groupe d'auteurs.

- WikiNi offre un moyen particulièrement simple, efficace et rapide de créer et gérer un site internet ou intranet. Il s'installe en une dizaine de minutes sur un serveur web supportant Php et une base de données MySQL. Cet outil permet, en ligne, avec n'importe quel navigateur Web :
  - de créer, supprimer, modifier, commenter les pages d'un site, quel que soit le nombre d'éditeurs et de pages.
  - de gérer les droits d'accès aux différentes pages (lire, écrire, commenter).
  - d'élaborer la mise en page des contenus de manière intuitive et très visuelle, par des règles de formatage ne nécessitant aucune connaissance informatique.
  - de publier instantanément toute création ou modification de page.
  - d'analyser, de gérer l'ensemble du site à partir de fonctions simples : plan du site, liste des utilisateurs, suivi des dernières pages modifiées ou commentées, etc.

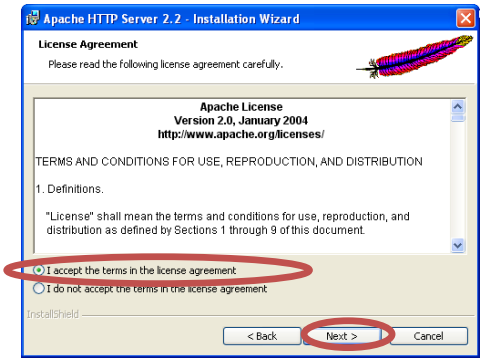
8 Installation et configuration d'un serveur Apache sous Windows XP/Vista/7 (Optionnel)

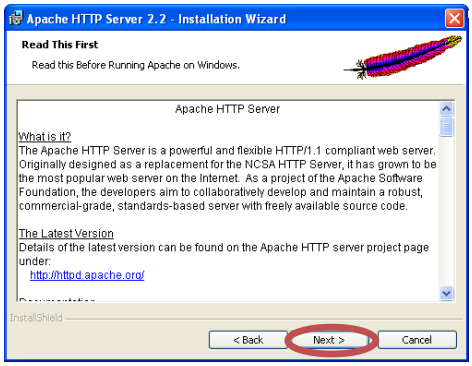
Commencez par télécharger apache http server version 2.2.19 avec OpenSSL ou la version 2.2.19 sans le module ssl.

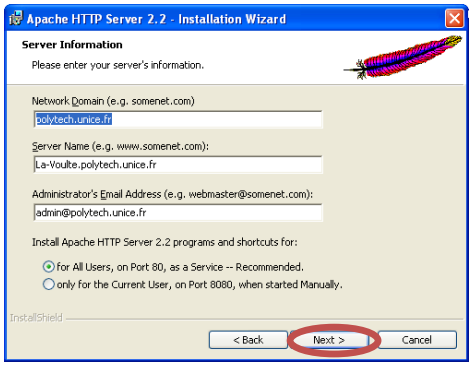
Avant toute chose, le protocole TCP/IP doit être installé et fonctionnel. Si vous possédez Windows NT 4 il est fortement recommandé que le Service Pack 6 soit installé.

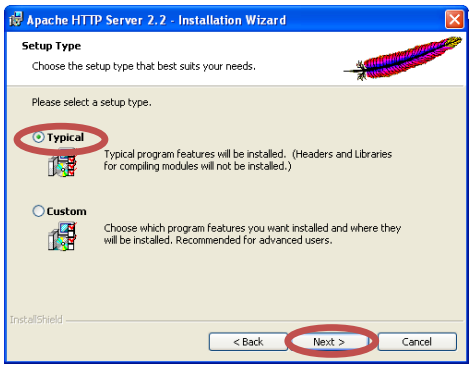
8.1 Installation

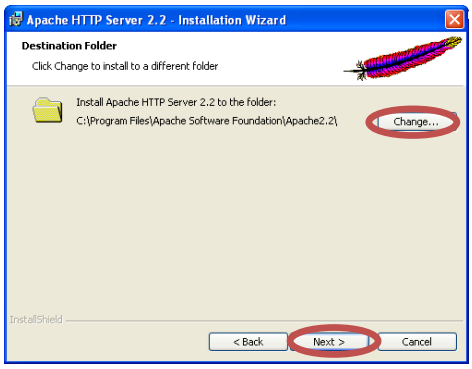
Cliquez deux fois sur le fichier *msi* et suivez les instructions (voir si dessous pour les choix à faire en fonction des différents écrans). Pensez bien à remplacer les identifiants donnés comme exemple par votre propre identifiant.

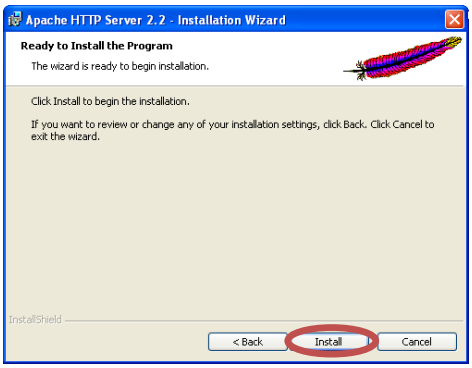


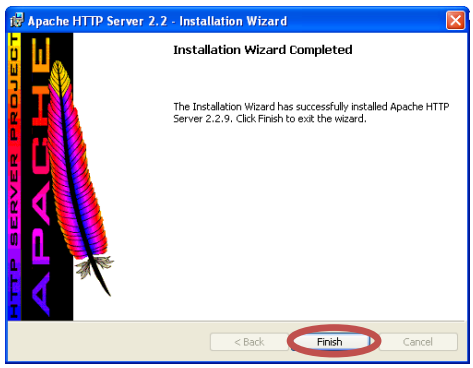












Bravo vous avez fini l'installation d'apache http Server 2.2. Pour vérifier que tout va bien ouvrez la page suivante depuis un navigateur se trouvant sur la même machine que le serveur apache :

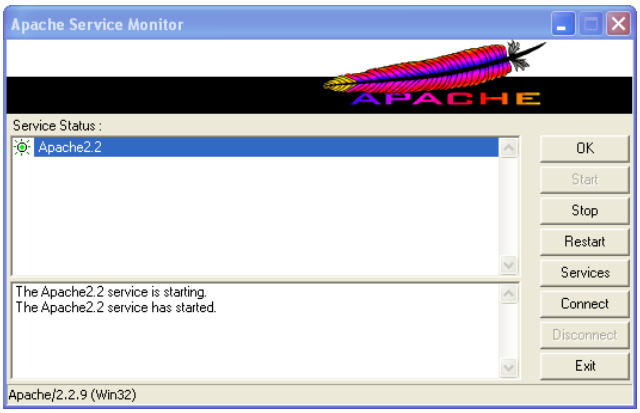
`http://localhost/`

8.2 Configuration

Concernant la configuration du serveur, tout fonctionne comme sous linux. Si vous avez installé Apache http server dans le répertoire *C:\Program Files\Apache Software Foundation\Apache2.2*

- Vous trouverez les fichiers de configurations dans le répertoire *C:\Program Files\Apache Software Foundation\Apache2.2\conf*. Notez que tout est configuré dans le fichier *httpd.conf*.
- Que la racine du serveur web (là où les pages sont stockées) est le répertoire *C:/Program Files/Apache Software Foundation/Apache2.2/htdocs*.
- La commande pour démarrer le serveur est *"C:\Program Files\Apache Software Foundation\Apache2.2\bin\httpd.exe" -w -n "Apache2.2" -k start*
- La commande pour redémarrer le serveur est *"C:\Program Files\Apache Software Foundation\Apache2.2\bin\httpd.exe" -w -n "Apache2.2" -k restart*
- La commande pour arrêter le serveur est *"C:\Program Files\Apache Software Foundation\Apache2.2\bin\httpd.exe" -w -n "Apache2.2" -k stop*

En plus de cela, vous bénéficiez sous Windows de l'Apache *Service Monitor*. Une petite interface d'administration bien pratique.



8.3 A vous de jouer

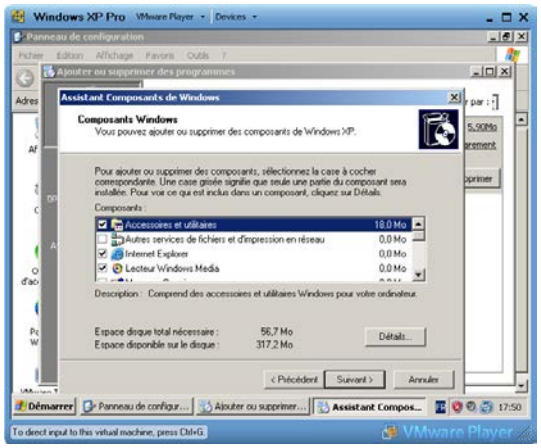
De la même façon que sous linux, configurez un serveur virtuel (virtual Host) et protégez en l'accès (accès limité à l'utilisateur *durant* qui a le mot de passe *tnarud*).

9 Installation et configuration d'un serveur IIS sous Windows (Optionnel)

Ne Pouvant pas vous distribuer à chacun un CD de windows XP, nous ne pourrons pas vous faire installer IIS. Cependant voici la démarche suivie (très simple) que vous pourrez reproduire sur votre propre machine.

9.1 Installation

Allez dans le panneau de configuration de Windows puis choisissez «ajouter ou supprimer des programmes » puis allez dans « Ajouter ou supprimer des composants Windows » dans le menu de gauche. Vous arriverez alors sur l'écran suivant :



Sélectionnez IIS dans la liste déroulante (cochez la case) puis allez dans Détails. Là, vous pouvez ajouter ou supprimer des options (comme le service SMTP ou la fonction FTP ...). Faites OK puis suivant. L'installation se termine tranquillement. Ouvrez un navigateur dans la machine virtuel et allez à l'adresse suivante pour vérifier que votre serveur http fonctionne correctement : http://localhost/.
Vérifiez également les points suivants :

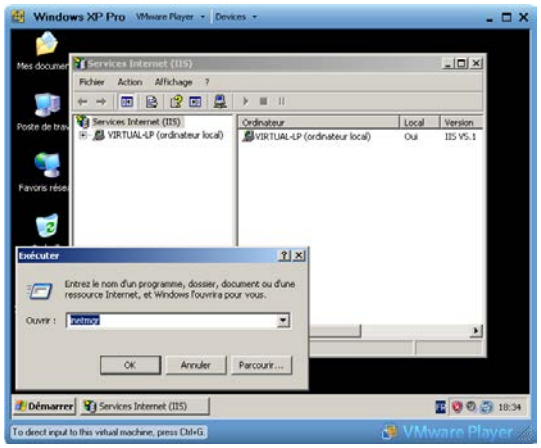
- L'installation de votre serveur IIS, a automatiquement installé à la racine de votre disque local "C", un dossier nommé "Inetpub", lequel est le dossier racine de votre serveur "ISS".
- Ce dossier racine "Inetpub", contient bien plusieurs dossiers de fichiers, dont un dossier nommé "wwwroot (le chemin du dossier ou répertoire nommé "wwwroot" est bien C :\Inetpub\wwwroot\)

9.2 Les virtuals Hosts

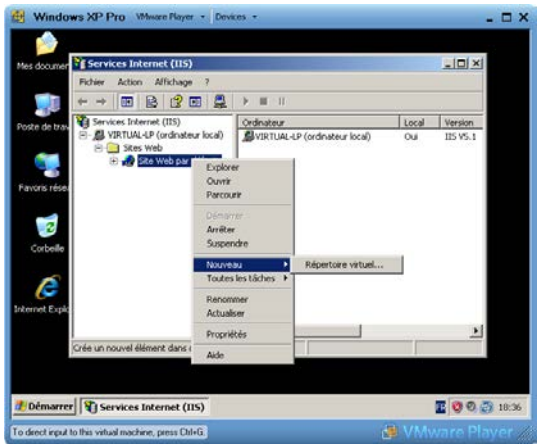
Allez dans le dossier *Inetpub/wwwroot* qui contient déjà par défaut, plusieurs fichiers, dont certains portent l'extension ASP (de : Active Server Page). La page dynamique localstart.asp est une page de démarrage, pour ce dossier *wwwroot*.
Ouvrez le fichier "localstart.asp ", contenu dans le répertoire racine, *wwwroot*. La page que vous afficherez, comporte des éléments vous indiquant très grossièrement le fonctionnement du serveur IIS que vous venez d'installer. Cependant, et avant de refermer cette page, notez bien la structure de son chemin complet dans la zone de liste déroulante de votre navigateur Internet :

```
C :\Inetpub\wwwroot\localstart.asp
```

Cette structure de chemin, vous aidera à comprendre la signification précise de la syntaxe du répertoire racine, lorsque vous créerez vos premières pages web dynamiques avec leurs extensions spécifiques ASP et/ou ASPX.
Exécutez la commande *inetmgr* pour ouvrir et accéder à votre serveur IIS.

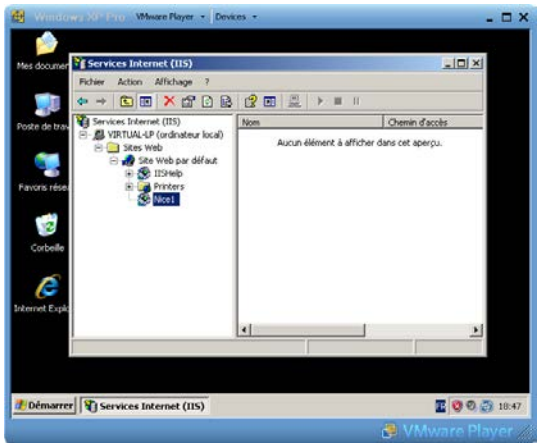


Déroulez, dans la fenêtre de votre serveur IIS, le nœud du dossier *Sites web*, pour faire apparaître la dépendance *Site Web par défaut*, sur laquelle vous appliquerez un clic droit, puis vous sélectionnerez la commande "Nouveau->Répertoire virtuel..."

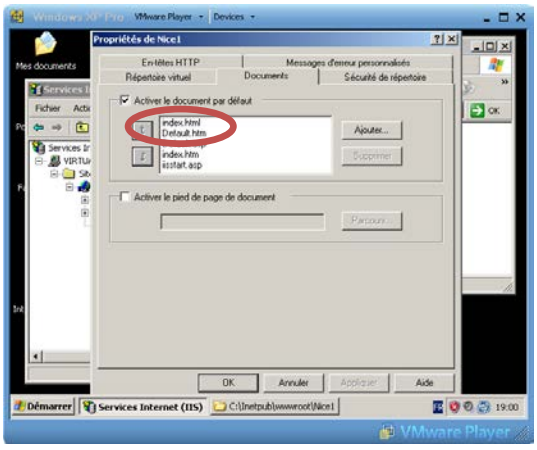


Donnez un nom (alias) à votre répertoire virtuel, lequel va représenter le nom de votre dossier de site racine, qui sera en vérité, le même que celui que portera le nom de votre répertoire de site web virtuel. Par la suite, nous ferons l'hypothèse d'avoir choisi Nice1 comme nom. Faites suivant puis cliquez sur le bouton "Parcourir" afin de créer votre répertoire de site web virtuel et le placer comme sur l'image ci-dessous.

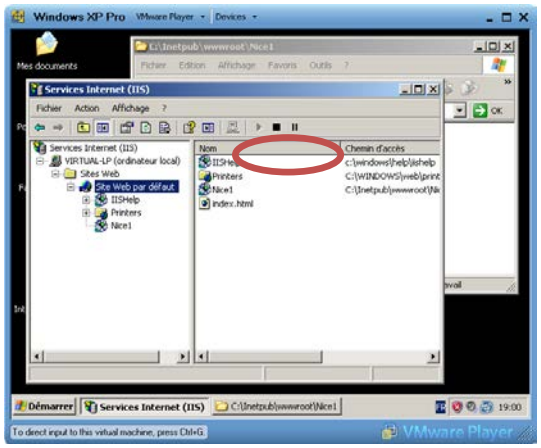
Cliquez sur "Suivant", après avoir vérifié que l'adresse du répertoire de contenu de votre site web est bien C :\Inetpub\wwwroot\Nice1. Laissez les autorisations et paramètres d'accès par défaut tels quels ; ils conviendront pour la plupart des cas, puis cliquez sur "Suivant" pour continuer. Quittez cet assistant afin d'afficher la structure de votre nouveau répertoire virtuel.



Copier un fichier index.html dans votre nouveau répertoire. Si vous vous rendez à l'url http://xxx.xxx.xxx.xxx/nice1/ vous ne verrez qu'un message d'erreur s'afficher. Les fichiers par default étant différents sous IIS. Faites un clic droit->propriété sur Nice1 dans votre interface de contrôle puis allez dans l'onglet Document. Ajoutez index.html à la liste.



Nous devons redémarrer le serveur pour que les modifications soient prises en compte. Faites-le à l'aide de l'interface de contrôle comme indiqué sur la figure suivante.



9.3 A vous de jouez

Explorez les différentes options de l'interface de contrôle pour gérer les droits d'accès (comme les .htacces). De la même façon que sous apache, configurez un serveur virtuel (virtual Host) et protégez en l'accès (accès limité à l'utilisateur *durant* qui a le mot de passe *tnarud*).

10 Quelques éléments complémentaires (optionnel)

10.1 Google analytics

http://www.webrankinfo.com/dossiers/google-analytics/trucs-et-astuces

http://www.webrankinfo.com/dossiers/google-analytics/management-api

http://www.webrankinfo.com/dossiers/google-analytics/api-nouveautes-sept-09

10.2 Google adresses

http://www.google.com/local/add/analyticsSplashPage?message=qbwelcome&hl=fr&gl=FR

10.3 Position d'un mot de passe sur google

http://www.positeo.com/check-position/