Cours de service réseaux

ISIMA F5 ZZ2

Christophe Gouinaud

Table des matières

1 Introduction et Historique **3**

2 Modèle des années 2000 **3**

3 Service web **4**

3.1 Protocole http 4
3.2 Démon WEB 9
3.2.1 Apache 10
3.2.2 IIS 12
3.2.3 autre et proxy 14
3.3 Clients WEB 14
3.3.1 Netscape 14
3.3.2 Internet Explorer 15
3.3.3 Autres 15
3.4 Contenus statiques 15
3.4.1 Texte HTML 15
3.4.2 Image GIF/JPEG 16
3.5 Page dynamique 17
3.5.1 Théorie 17
3.5.2 CGI 17
3.5.3 PHP 26
3.5.4 Javascript 35

1 Introduction et Historique

Les applications sur réseau informatique jouent en permanence avec les modes de répartition des données entre terminaux et oridnateurs centraux. Les années 80 ont vu la généralisation des serveurs de fichiers, les années 90 le triomphe du modèle client-serveur et les années 00 commencent avec le client léger et plus particulièrement les applications utilisant le transport http.

2 Modèle des années 2000

Cette partie devrait être intitulée *que pourrait-on faire avec tout cela ?*. Dans l'historique précédent nous constatons une constante oscillation du positionnement des différentes fonctions du traitement de l'information entre les postes près du client et les ordinateurs centralisés. Les raisons de cette valse hésitation sont souvent liées à des modes d'implémentation imposés par l'optimisation du quadruplé : coût de développement, coût de déploiement, coût de maintenance, coût de formation des utilisateurs.

Durant les années 80 et 90, le personnel des entreprises est devenu de plus en plus précaire, les applications de l'informatique se sont multipliées, ce qui a conduit à une nécessité de réduire les temps de déploiement et de formation propres à chaque application. Ceci a donc conduit à une tentative d'unification des interfaces et à une centralisation accrue des fonctions de traitement.

Le développement des réseaux extensifs à débit aléatoire a aussi conduit les industriels de l'informatique à remettre en question les modèles d'interfaces, de type XWindows, reposant sur des protocoles connectés à états.

Le développement du web et de ces diverses extensions ont apporté une réponse presque parfaite à ces différentes contraintes. En effet, les interfaces à base de langage conpréhensible par les clients web (prononcer cela browser) présentent l'avantage d'utiliser des standards compréhensibles par tous et d'une ergonomie indépendante de l'application. Par exemple, quelque soit la page web que vous regardez, vous utilisez le bouton de la souris dont vous avez l'habitude pour dérouler un menu ou cocher un choix. Cette facilité fait qu'il n'est plus nécessaire de former le personnel à l'ergonomie propre du système.

Cette solution est aussi très satisfaisante pour le programmeur dans la mesure où il n'a plus à ce préoccuper de la plateforme finale d'utilisation de son application s'l respecte au mieux les standards du web [1]. Dans ce même cas, les coûts de déploiement et de maintenance d'une application sont réduits au minimum du fait de l'extrême centralisation de l'application. La gestion de l'obsolescence des versions est également extrêmement simple car elle ne pose pas le problème de l'arrêt de l'application durant la période de déploiement.

Un autre avantage induit par la centralisation des traitements est que l'accès aux données de l'application se fait en un seul point et donc permet l'utilisation simplifiée de fichiers. Cela dit la plupart des développeurs s'oriente vers une utilisation conjointe d'une base de données relationnelle qui fournit un cadre simplifié pour la gestion des données.

Le problème principal posé par ce type d'applications est une sensibilié à la monté en charge de l'application qui peut conduire à des situations de blocage des serveurs. L'abscence de notion d'état du protocole http conduit aussi le programmeur à prendre en charge lui-même la notion de sessions, ce qui est inhabituel du fait qu'en programmation classique, la session est fournie par le système d'exploitation. L'utilisation du protocole http pose aussi plusieurs problèmes de sécurité dans la mesure où il a été conçu pour être utilisé sans identification et sans cryptage. Ce dernier aspect devra donc être géré par le développeur au lieu que ce soit le système d'exploitation qui en ait la charge.

Restait le problème du transfert de fichiers et d'informations (fichier texte, fichier de tableur, image, graphique) produit par des applications embarquées sur les postes. Le mode d'échange de ce type de données privilégiées des utilisateurs est souvent le mail. Une analyse poussée de cette situation montre que cette situation est bien souvent le résultat d'une mauvaise organisation de l'espace disque et des propriétés des fichiers dans l'entreprise. L'aspect transfert et sécurité étant géré par le serveur de mail, l'utilisateur a peu d'opérations à réaliser. Malheureusement, cette méthode génère une surconsommation d'espace disque, une multiplication des versions pour un même fichier et une bordélisation du flux d'information de l'entreprise. Elle introduit de plus une délégation implicite de sécurité, facilite la propagation de virus et mélange les différents types d'information. Elle ne permet pas non plus de sauvegarde simple des différentes informations en fonction de leur criticité. Bref, c'est du bricolage.

[1] Ceci admet quelques bémols du fait des différences d'interprétation du code HTML par les browsers mais peut être très fortement limité en faisant les choses simplement

FIG. 1 – Transaction normale sur le WEB

3 Service web

Les services web sont basés sur le protocole http inventé au CERN en 1992. Ce protocole conçu à l'origine pour la documentation a connus le succès que l'on sait et a permis la popularisation d'Internet du fait de la simplicité de l'utilisation des logiciels qui l'exploitent. L'apparition d'extensions logicielles propres à réaliser des applications complexes n'a pas tardé et a permis le développement du commerce en ligne, puis d'applications de gestion d'entreprise.

Ces fonctions sont assurées pour l'essentiel par des scripts lancés par la partie serveur (appelé démon WEB dans la suite) associé à d'eventuels programmes tournant à travers la partie cliente (appelé browser dans la suite). Du CGI[2], on est passé rapidement à des scripts intégrés au démon WEB tels php ou asp. Pour le côté browser, on est passé du simple affichage de text enrichi (HTML) d'images et de liens hypertexts à des langages complets tel que javascript ou java. Nous allons dans cette partie étudier en détail ces divers objets.

3.1 Protocole http

Une transaction WEB correspond à un seul échange de données entre le client et le serveur à travers une socket tcp. Pratiquement, les choses se passent ainsi :

- Le client provoque l'ouverture d'une socket tcp sur un port convenu du serveur (80 par défaut).
- Le serveur accepte la demande de connexion tcp puis se met en lecture sur la socket
- Le client écrit dans la socket sa demande
- Le serveur expédie sa réponse, assortie d'un code d'opération
- Une fois toutes les données reçues, le client ferme la connexion tcp

Une fois la transaction terminé, le client examine le résultat de la requète et demande les éventuels documents associés. Cela signifie qu'une page WEB comportant une image gif sera chargé en deux transactions. L'ensemble du résultat est ensuite *affiché* par le client.

Ce mécanisme peut paraître lourd et contraignant mais permet en fait de n'échanger au cours d'une transaction que des documents d'un seul type et par conséquent évite de recourir à un codage complexe des échanges. Elle permet également de répartir les fichiers à expédier sur des serveurs différents de façon simple.

with two servers : An HTTP server running on

La figure 3.1 montre un client Web (A) tournant sur un pc qui va réaliser une transaction avec un serveur qui fait tourner un serveur Web (B) puis une transaction avec un deuxième serveur (C).

HTTP server running on Computer B. This document Resource Locator (URL) for that link might look

Le client WEB tournant sur A récupère un document qui était stocké dans un fichier nommé docu1.html. Cette page WEB contient un lien vers un autre document stocké sur C dans le fichier docu2.html. L'URL[3] pour ce lien est :

```
http://C.isima.fr/docu2.html
```

Rappelons ici que les URL ont pour syntaxe :

```
PROTOCOLE://HOST:PORT/CHEMIN/FICHIER
```

ou :

- PROTOCOLE est le protocole à employer pour contacter le serveur, typiquement http, ftp, shttp ...
- HOST est son adresse Internet
- PORT est le port à utiliser et n'est à spécifier que s'il est différent de celui défini en standart (80 :http ...)

[2] Common Gateway Interface
[3] Uniform Ressource Locator

– CHEMIN/FICHIER est le nom complet du fichier dans l'arborescence des documents du serveur web. En général, il correspond à un nom à partir d'un répertoire qui est la racine des documents du serveur web. Il faut aussi remarquer que l'on utilise toujours des / pour séparer les répertoires et sous répertoires, quelque soit le système d'exploitation. Le daemon serveur WEB doit en assurer la traduction.

Computer C and displays it on the monitor connected to

Si l'utilisateur de A clique sur le lien, son broswer contacte le serveur http de C, récupère le fichier docu2.html et affiche le résultat sur l'écran de A.

following example shows what an HTTP client fetches docu2.html.

Le protocole http spécifie la communication entre le serveur et le client. En clair, ce protocole donne le format du bloc de donnée que le client doit envoyer au serveur une fois la connexion tcp ouverte. Dans la pratique, le client envoie sa requète sous la forme suivante :

```
GET /docu2.html HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent:  Lynx/2.2  libwww/2.14
From:  gouinaud@isima.fr
     * a blank line *
```

version 1.0 to communicate. The client also lists the as a Lynx client. (The "Accept :" list has been truncated

La première ligne indique le type de requête, GET ici, le fichier requis, /docu2.html, et le protocole à utiliser. Ensuite les lignes Accept : spécifie les formats de données que le client peut accepter. Il s'agit de types MIME[4] tels que l'on peut les trouver également dans les mails. Puis le client dit par la ligne User-Agent : quel type de browser il est, Lynx version 2.2 ici, puis il dénonce son utilisateur à l'aide du champ From :.

La fin de la requète est spécifiée par un retour chariot et le client se met alors en écoute de la réponse sur la socket tcp.

La réponse du serveur est de la forme suivante :

```
HTTP/1.0 200 OK
Date: Wednesday, 02-Feb-94 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Monday, 15-Nov-93 23:33:16 GMT
Content-type: text/html
Content-length: 2345
     * a blank line *
<HTML><HEAD><TITLE> . . . </TITLE> . . .etc.
```

200 indicating it has successfully processed the indicates it is using MIME version 1.0 to describe the "Content-type :" header. Finally, it sends the number of

Par la première ligne de ce message, le serveur accepte l'utilisation du protocole http 1.0 pour le dialogue et indique par le numéro 200 qu'il peut satisfaire la demande du client. Il donne ensuite la date, puis indique le nom et la version du serveur employé et la version de Mime qu'il connait. Les lignes suivantes indiquent la dernière date de mise à jour du document, le type du contenu, text/html ici, la longueur du document et enfin un retour chariot pour signifier que le header est terminé. Il ecrit ensuite dans la socket le contenu du document qu'il aura éventuellement traduit dans un format acceptable par le client. Quand le document est terminé c'est-à-dire que le client a reçu le nombre d'octets correspondant à la requète, il ferme la connection tcp sans qu'aucun autre échange n'ait lieu.

Nous pouvons remarquer ici que :

data into a form the client can accept.

– Le client et le serveur utilisent des headers identiques au email au sens de la norme RFC 822.

[4]Multipurpose Internet Mail Extension

– Le client peut annoncer autant de formats accept qu'il souhaite, le serveur étant chargé de traduire ces documents dans l'un de ces format, ou de renvoyer une erreur en cas d'impossiblité.
– A aucun moment, un mot de passe n'a été requis dans cette transaction

Il nous faut maintenant examiner les différentes méthodes disponibles pour ces transactions. Il en existe trois : GET, HEAD et POST en http 1.0 mais des serveurs et clients peuvent en connaitre plus. Leur description détaillée est la suivante :

– GET
  La méthode GET est employé pour demander des données à un serveur en utilisant un URL. Si URL référence un programme du serveur, c'est bien evidemment le résultat de l'exécution du processus par le serveur qui sera retourné. Les paramètres seront fournis dans le bloc de données de la commande GET.
  La commande GET peut être aussi conditionnelle si le header de la requète comporte un champ If-Modified-Since. Le serveur ne doit dans ce cas envoyer les données que si la ressource a été modifié à une date plus récente que celle qui est fournie. Il s'agit bien évidemment ici de réduire le trafic réseau engendré par le service.
– HEAD
  Cette méthode est identique à la méthode GET du point de vue syntaxe. La réponse du serveur sera constituée de codes d'acceptation et des éventuels meta champ du document. En gros, le corps du document ne sera pas tramsmis par le serveur, seul l'entête et le type de document le seront. Cette méthode sert à testé la validité d'un URL et elle permet l'indexation de document dont les champ META ont été remplis. Il n'y a pas de head conditionnel, l'éventuel champ If-Modified-Since sera toujours ignoré par le serveur.
– POST
  La méthode POST est utilisée pour envoyer des données complexes à un serveur WEB. Il s'agit de la méthode permettant de modifier le contenu du serveur et non plus simplement d'aiguiller des choix parmi les documents qu'il propose : Les utilisations principales en sont :
  – Ajoute de l'information à des ressources existantes. Par exemple, ajoute un article dans un forum/
  – Envoyer des données à un serveur en vue de leur traitement
  – Enrichir une base de données

  Lors de l'utilisation de la méthode POST la configuration du serveur decidera de quel programme doit être appelé pour traiter les données que vous soumettez. Le client ne doit lui se préocupper que du format des données qu'il soumet et non du type de programme qui les traitera. Cette caracteristique introduit une forte indépendance entre l'interface et le traitement et donc une grande souplesse pour le gestionnaire du site.
  Le navigateur de l'utilisateur considérera que son action est réussie à partir du moment où il recevra la réponse du serveur, qui peut être soit le code 200 (ok) avec un contenu à afficher, soit 204 (no content) s'il n'y a rien de plus à afficher. L'envoi de ces codes ne présuppose pas que le programme chargé de traiter ces données l'a fait correctement. Il est donc de bon aloi d'envoyer une page HTML indiquant le succès ou l'échec du traitement à l'utilisateur de façon que celui-ci puisse corriger ses éventuelles erreurs.
  Si l'envoi visait à créer un nouveau fichier sur le serveur, le code 201 peut être retourné avec un contenu en HTML contenant un pointeur vers la nouvelle ressource.
  Les contraintes d'une opération POST sont de fournir une query-string dans un format correct et un Content-Length valide dans l'entête de la requête. Si le serveur ne peut déterminer la taille du message, il retournera une erreur 400 indiquant que la requête est mal formulée, ceci afin de se protéger contre les requêtes visant à provoquer des dépassements de pile. **Un programme chargé de répondre à ce type de requête devra toujours tester l'ajustement de la taille des données aux tailles qu'il peut accepter pour ne pas mettre en péril l'application et la machine sur laquelle elle tourne**.
  Les navigateurs ne doivent jamais cacher les réponses aux requêtes POST, car rien ne permet de distinguer dans l'URL deux POST successifs contenant des données différentes. Le non respect de cette consigne peut entraîner des confusions pour l'utilisateur et comporte des risques de fuite accidentelle d'information.

Voici la description détaillée des erreurs d'après la RFC en anglais.

Informational 1xx	This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. HTTP/1.0 does not define any 1xx status codes and they are not a valid response to a HTTP/1.0 request. However, they may be useful for experimental applications which are outside the scope of this specification.
Successful 2xx	This class of status code indicates that the client's request was successfully received, understood, and accepted.
200 OK	The request has succeeded. The information returned with the response is dependent on the method used in the request, as follows : GET an entity corresponding to the requested resource is sent in the response ; HEAD the response must only contain the header information and no Entity-Body ; POST an entity describing or containing the result of the action.
201 Created	The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the entity of the response. The origin server should create the resource before using this Status-Code. If the action cannot be carried out immediately, the server must include in the response body a description of when the resource will be available ; otherwise, the server should respond with 202 (accepted). Of the methods defined by this specification, only POST can create a resource.
202 Accepted	The request has been accepted for processing, but the processing has not been completed. The request may or may not eventually be acted upon, as it may be disallowed when processing actually takes place. There is no facility for re-sending a status code from an asynchronous operation such as this. The 202 response is intentionally non-committal. Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent's connection to the server persist until the process is completed. The entity returned with this response should include an indication of the request's current status and either a pointer to a status monitor or some estimate of when the user can expect the request to be fulfilled.
204 No Content	The server has fulfilled the request but there is no new information to send back. If the client is a user agent, it should not change its document view from that which caused the request to be generated. This response is primarily intended to allow input for scripts or other actions to take place without causing a change to the user agent's active document view. The response may include new metainformation in the form of entity headers, which should apply to the document currently in the user agent's active view.
Redirection 3xx	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The action required may be carried out by the user agent without interaction with the user if and only if the method used in the subsequent request is GET or HEAD. A user agent should never automatically redirect a request more than 5 times, since such redirections usually indicate an infinite loop.

300 Multiple Choices	This response code is not directly used by HTTP/1.0 applications, but serves as the default for interpreting the 3xx class of responses. The requested resource is available at one or more locations. Unless it was a HEAD request, the response should include an entity containing a list of resource characteristics and locations from which the user or user agent can choose the one most appropriate. If the server has a preferred choice, it should include the URL in a Location field ; user agents may use this field value for automatic redirection.
301 Moved Permanently	The requested resource has been assigned a new permanent URL and any future references to this resource should be done using that URL. Clients with link editing capabilities should automatically relink references to the Request-URI to the new reference returned by the server, where possible. The new URL must be given by the Location field in the response. Unless it was a HEAD request, the Entity-Body of the response should contain a short note with a hyperlink to the new URL. If the 301 status code is received in response to a request using the POST method, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued. Note : When automatically redirecting a POST request after receiving a 301 status code, some existing user agents will erroneously change it into a GET request.
302 Moved Temporarily	The requested resource resides temporarily under a different URL. Since the redirection may be altered on occasion, the client should continue to use the Request-URI for future requests. The URL must be given by the Location field in the response. Unless it was a HEAD request, the Entity-Body of the response should contain a short note with a hyperlink to the new URI(s). If the 302 status code is received in response to a request using the POST method, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued. Note : When automatically redirecting a POST request after receiving a 302 status code, some existing user agents will erroneously change it into a GET request.
304 Not Modified	If the client has performed a conditional GET request and access is allowed, but the document has not been modified since the date and time specified in the If-Modified-Since field, the server must respond with this status code and not send an Entity-Body to the client. Header fields contained in the response should only include information which is relevant to cache managers or which may have changed independently of the entity's Last-Modified date. Examples of relevant header fields include : Date, Server, and Expires. A cache should update its cached entity to reflect any new field values given in the 304 response.
Client Error 4xx	The 4xx class of status code is intended for cases in which the client seems to have erred. If the client has not completed the request when a 4xx code is received, it should immediately cease sending data to the server. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. Note : If the client is sending data, server implementations on TCP should be careful to ensure that the client acknowledges receipt of the packet(s) containing the response prior to closing the input connection. If the client continues sending data to the server after the close, the server's controller will send a reset packet to the client, which may erase the client's unacknowledged input buffers before they can be read and interpreted by the HTTP application.

400 Bad Request	The request could not be understood by the server due to malformed syntax. The client should not repeat the request without modifications.
401 Unauthorized	The request requires user authentication. The response must include a WWW-Authenticate header field (Section 10.16) containing a challenge applicable to the requested resource. The client may repeat the request with a suitable Authorization header field (Section 10.2). If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user should be presented the entity that was given in the response, since that entity may include relevant diagnostic information. HTTP access authentication is explained in Section 11.
403 Forbidden	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request should not be repeated. If the request method was not HEAD and the server wishes to make public why the request has not been fulfilled, it should describe the reason for the refusal in the entity body. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.
404 Not Found	The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. If the server does not wish to make this information available to the client, the status code 403 (forbidden) can be used instead.
Server Error 5xx	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request. If the client has not completed the request when a 5xx code is received, it should immediately cease sending data to the server. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These response codes are applicable to any request method and there are no required header fields.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501 Not Implemented	The server does not support the functionality required to fulfill the request. This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.
502 Bad Gateway	The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
503 Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition which will be alleviated after some delay. Note : The existence of the 503 status code does not imply that a server must use it when becoming overloaded. Some servers may wish to simply refuse the connection.

Les sources de données sont citées, le lecteur intéressé par plus de détail pourra se reporter au site du W3C (http ://www.www.org) et notamment aux RFC (rfc1945 par exemple).

3.2 Démon WEB

Il ne s'agit pas ici de décrire toutes les fonctions des démons implémentant le protocole http, mais de donner les repères utiles à l'administrateur d'application ou de serveur WEB.

3.2.1 Apache

Apache est le démon WEB le plus populaire à l'heure où j'écris ces lignes. Il s'agit d'un logiciel freeware que tout un chacun peut se procurer (www.apache.org) , soit sous forme de code source, soit sous forme de binaire exécutable. Apache représente 56 % du marché des serveurs web.

A. *Installation*

Pour faire une installation d'Appache, il faut récupérer la distribution à http ://www.apache.org/ sous forme binaire ou sous forme de code source. Si vous choisissez cette dernière solution, il vous faudra :

- Dérouler le fichier tar dans un répertoire
- Taper ./configure en choisissant le répertoire d'installation /usr/local/apache par défaut
- Lancer la compilation et l'installation en tapant make install
- Lancer le daemon /usr/local/apache/sbin/apachectl start
- Copier vos fichiers html dans le répertoire ad hoc et votre serveur est prêt.

Cette procédure simple s'applique quand vous n'avez pas de desirata particulier par rapport à l'utilisation de modules non inclus dans la distribution, php par exemple, et quand les répertoires par défaut vous conviennent. Ce n'est bien évidemment que rarement le cas.

Il faut distinguer deux types de configuration : celles qui se font avant la compilation et celles qui se font après. La commande ./configure –help donne la liste des options de conf.

A titre d'exemple, voici la procédure permettant d'inclure un support php natif par Apache :

- Récupérer la distribution de php à http ://www.php.net/
- Dérouler la distribution php
- Dérouler la distribution Apache
- Configurer php ./configure dans le répertoire de la distribution avec la directive with apache et en précisant le directory d'Apache
- Configure Apache
- Compiler et installer php avec make install
- Compiler et installer Apache

Cette procédure peut encore se compliquer si on ajoute d'autres modules ou si les modules ajoutés dépendent d'autres applications. Par exemple, php+ldap+mysql+apache.

La partie suivante traite de la configuration après installation.

B. *Configuration.*

La configuration d'Apache repose uniquement, depuis la version 1.3.2, sur le fichier http.conf qui se trouve par défaut dans /usr/local/apache/conf

Voici quelques options du fichier de configuration :

- Gestion des répertoires

ServerRoot DIR	répertoire le plus haut des fichiers du serveur
DocumentRoot "DIR"	répertoire des documents
User_dir NAME	répertoire perso des utilisateurs
ScriptAlias /cgi-bin/ DIR	répertoire des scripts apparaissant comme cgi-bin

Après ces définitions d'ordre général, chaque directory devra être configuré précisément, pour les types de fichier reconnus ou utilisables et les conditions d'accès.

Voila l'exemple du minimum pour un directory public :

```
<Directory "/home/gargamel/web">

# lien symbolique, Index et SSI autorise
    Options Indexes FollowSymLinks MultiViews IncludesNoExec

#  le fichier ACL, si il existe à le dernier mots sur l'accès
    AllowOverride None

#
#  Pas de controle
#
```

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

Voila l'exemple pour un répertoire complètement sécurisé :

```
<Directory /home/gargamel/web/intra>

Options Indexes FollowSymLinks IncludesNoExec

AllowOverride None

order deny,allow
deny from all
allow from 193.54.49.29  asterix.univ-st-etienne.fr
AuthType Basic
AuthUserFile /usr/local/apache/conf/passwd
AuthName "Acces intra"
#require valid-user   #si y en plusieurs
require user gouinaud #si y en a qu'un
satisfy all
</Directory>
```

L'administrateur du serveur peut, bien évidemment, décliner toutes les combinaisons possibles de ces deux situations.

Le tuning du type de fichier autorisé et leur définition se fait à travers des AddType ou AddHandler dont les plus utiles sont les suivants :

application/x-httpd-php	correspond au script php
AddHandler cgi-script	exécution de commandes exécutables du système
AddHandler server-parsed	Server Side Include
AddHandler imap-file	Pour utiliser des images map

Toutes ces configurations sont à faire dans la section mime du fichier http.conf.

– Paramètres de fonctionnement du serveur :

PidFile	Fichier où est stocké le PID du serveur maître
StartServers	Nombre de serveurs à lancer
MaxClients	Nombre de clients maximum
Port	Port d'écoute
User	Login ou UID de l'utilisateur faisant tourner le serveur
Group	pareil, mais le groupe
ServerName	nom à renvoyer en cas de requête non qualifiée

Il est important de noter que ces paramètres influencent énormément la sécurité du serveur. En particulier, l'utilisateur du serveur devra être choisi de façon à limiter au maximum les droits du démon. En pratique, il vaut mieux le plus souvent utiliser le couple nobody,nogroup.

Pour tous ces paramètres, le fichier de configuration d'Apache fournit des valeurs adaptées à la plupart des cas.

Exercice (sous UNIX) :

1. Récuperer Apache et php dernière version
2. Compiler et installer le tout dans /usr/local
3. Votre machine de test n'étant pas dans le DNS, créer un fichier de hosts pour créer une correspondance pour son adresse IP et son nom de hosts et modifier son nom.
4. Créer un serveur web public dans le répertoire /home/inter. Ce serveur devra reconnaître index.html comme nom de fichier par défaut pour les directories, interdire l'affichage de répertoire vide et parser tous les fichiers html (pour les SSI par exemple).
5. Créer un intranet sur le port 6900 avec filtrage par adresse IP et par mot de passe. Sa racine devra être dans /home/intra/

6. Tester tout cela avec un autre poste sous divers système.
7. Modifier les scripts de démarrage de votre machine pour que vos serveurs soient accessibles même après un reboot.

3.2.2 IIS

A. *Introduction*

Cette suite de logiciels venant avec WNT Serveur ou W2000 serveur fournit trois types de service :

– Le service ftp, c'est-à-dire le transfert de fichiers bidirectionnel en mode texte ou binaire. Ce type de protocole est utilisable via un navigateur Web ou via la commande ftp. L'identification se fait par un couple login mot de passe avec possibilité éventuelle d'accès anonyme via le login anonymous.
– Le service http, c'est-à-dire le transfert de fichier type avec ou sans identification par site, par login mot de passe. Http permet également un fonctionnement en mode sécurisé via l'utilisation d'une couche de protocole appelée SLL.
– Le système gopher, ancêtre du précédent maintenant abandonné.

IIS prend en charge ces trois types de service avec plus ou moins de bonheur. En effet, ces protocoles et outils ont été conçus pour des machines UNIX et des points de non compatibilité peuvent apparaître.

IIS permet de lancer rapidement un service WEB ou ftp comportant des zones sécurisées. Il est par contre très consommateur de ressources et devra donc être déployer sur des machines de puissance importante dans le cas de serveur populaire.

B. *Installation*

C'est une installation classique Microsoft :

– Aller dans le répertoire /I368/inetsvc/
– Lancer inetstp.exe

Tant que l'on y est, on installe frontpage qui permet de gérer et de dessiner les pages des sites.

L'installation crée plusieurs groupes de programme :

– Internet Information Server
  Ce groupe contient :
  + le gestionnaire des services
  + le gestionnaire de clefs
  + un accès à la doc en ligne
  + un gestionnaire d'accès via HTML

  Tous ces élément permettent une gestion simplifiée du service WEB. Via ce menu, l'accès à la doc en ligne se fait par le service de fichiers et non WEB. Ceci permet d'accéder à la doc même quand le serveur WEB disfonctionne.
– Microsoft Frontpage
  Ce groupe contient les éléments et les programmes du logiciel Frontpage.
  Principalement, Frontpage editor et Frontpage explorer.

Outre l'installation des programmes et les inscriptions en base de registres, cette installation fait deux choses principales :

– Enregistrer le service auprès du gestionnaire de services (à voir avec panneau de conf service)
– Créer un utilisateur anonyme IUSR_HOSTNAME qui servira à l'accès anonyme au service ftp et au service http. Il faut être très prudent dans la gestion de ce login sous peine de bloquer tout le service WEB.

Le lecteur attentif en déduira la méthode pour démarrer et arrêter le service. Tester son arrêt, son démarrage et son bon fonctionnement via votre navigateur.

Il arrive que l'on conçoive un serveur HTTP avant de disposer d'une entrée DNS ou sur une machine différente de celle qui publie le serveur. Afin de pouvoir tester ses pages en utilisant le nom de la machine plutôt que son adresse IP, il faut pour cela créer une correspondance entre IP et nom de machine via le fichier */winnt/system32/driver/etc/hosts* .

Je vous conseille de le faire systématiquement en utilisant le même nom pour netbios et Internet.

C. *Administration*

L'administration d'un serveur WEB comprend différentes choses dont :

– La gestion des répertoires reflétant la structure du site
– La gestion de la politique d'autorisation d'accès aux pages du serveur

Les répertoires du serveur sont visibles en lançant l'outil d'administration du service IIS via le menu IIS.

Une fois lancé, il affiche la liste des machines du domaine exportant un service.

Le répertoire de base de votre machine, c'est-à-dire le http ://toto.wwwfc.fr/ correspond à la racine du serveur est /inetpub/wwwroot sauf si vous l'avez changé à l'installation.
Il vous faudra donc fixer ce répertoire sur celui qui correspondra à l'emplacement choisi pour les fichiers de votre site.

Exercice :

1. Créez un répertoire /htdocs sur la racine de votre disque et rendez le accessible en tant que répertoire racine.
2. Rendez accessible le répertoire d'origine microsoft en http ://toto.wwwwfc.fr/orig/
3. Mettez une page par défaut index.htm dans le répertoire et vérifiez que tout fonctionne.
4. Testez les droits d'accès nécessaires pour que les fichiers puissent être lus.
5. Faites en sorte que les pages par défaut de votre site s'appellent index.html

D. *Outils et services de gestion*

Un service NT peut-être arrêté, suspendu ou démarré. Dans le cas du WEB, la différence entre les deux (suspendu ou arrêté) est inexistante mais pour que le service fonctionne, il doit être bien évidemment démarré.
Le service de gestion comporte les onglets suivants :

– Service
  Définition du port de fonctionnement et des paramètres de connexion.
– Répertoire
  Définition des répertoires de travail, nous l'avons vu plus haut.
– Enregistrement
  Il concerne les logs du serveur Web.
– Avancé
  Permet de gérer les droit d'accès et autres.

Tous ces élément sont accessibles via l'interface de gestion WEB mais elle ne fonctionne pas toujours de façon optimum.

Exercice

1. Configurez votre machine pour accepter 2 connexions simultanées
2. Configurez votre machine pour que les logs soient accessibles dans /log via le service http.

La gestion des droits d'accès concerne plus particulièrement la création des serveurs d'intranet. Ce sont des serveurs de groupe de travail qui ont des restrictions d'accès. Un serveur de ce type est par exemple nécessaire pour diffuser une liste de prix à une agence commerciale via un serveur web qui est visible sur l'Internet.
On fait normalement tourner un serveur intranet sur un port différent du classique 80.
Il existe deux types de filtrage d'accès sur IIS :

– Le filtrage par adresse IP (pas par nom)
  Malheureusement celui-ci ne peut se faire que sur l'ensemble du service et non sur un simple répertoire. Le filtrage se fera par réseau ou machine via l'utilisation des netmask. Il a deux modes de fonctionnement : tout interdit et autorisation par réseau, ou tout autorisé et interdiction par réseau.
– Le filtrage par utilisateur NT
  Pour l'établir, il faut que le compte de l'utilisateur soit valide, que les fichiers que l'on veut protéger n'aient pas les droits d'accès en lecture pour Everybody et que l'utilisateur concerné ait un droit d'accès au moins en lecture sur le fichier. Il faut se méfier des expirations de mots de passe qui bloquent l'accès au service sans expliquer pourquoi.

Cette dernière fonctionnalité est une grande qualité d'IIS qui simplifie ainsi grandement la gestion des droit d'accès au serveur sans avoir besoin de gérer deux fichiers de passwd distincts.
Cette facilité se paye par un mini trou de sécurité car les passwd pour le service Internet circuleront sur le Net et par conséquent pourront être espionnés. Comme il s'agit des mêmes comptes que pour l'accès local, un vol de mot de passe aurait des conséquences graves. Il ne faut donc jamais utiliser le compte Administrateur pour accéder aux pages. Microsoft a bien prévu une circulation de mot de passe crypté mais malheureusement elle fonctionne mal avec la plupart des navigateurs.
Nous utiliserons donc sur le profil et les stratégies systèmes, pour limiter l'accès aux machines locales, des personnes qui ne font que de l'accès via le WEB.

Exercice :

1. Créez un intranet sur le port 90 de votre machine
2. Rendez cette intranet accessible à votre voisin de gauche mais pas celui de droite
3. Testez ...
4. Créez un utilisateur avec vos initiales
5. Mettez lui un mot de passe
6. Créez vous un login si ce n'est déja fait
7. Rendez un répertoire à son nom accessible par lui et vous
8. Testez en faisant bien attention à vider les caches de votre machine
9. Faites en sorte que vous ayez le droit de lire le contenu du répertoire mais pas lui.

E. *Conclusion*
Comme nous venons de le voir, IIS est d'un déploiement simple et rapide. Sa configuration est aisée et simple mais ses possibilités en terme d'intranet sont limitées au moins en ce qui concerne cette version. Si on désire une compatibilité plus avancée avec le monde UNIX, il faudra utiliser le serveur Apache dans sa version Windows.

3.2.3 autre et proxy

Il existe d'autres démons web qui ont des fonctionnalités similaires au précédent. Les différences sont souvent de l'ordre de la performance ou des conditions de mise en oeuvre.

Les daemon serveur http peuvent aussi jouer le rôle de proxy pour le protocole http. Ils se contentent alors de relayer les demandes qui leur sont faites en direction d'un autre serveur. Il ne faut pas les confondre avec les vrais proxies dont le rôle est de relayer les requêtes d'un réseau local vers un internet non accessible directement.

Il existe deux types de proxies :

– les proxies protocolaires,
Comme Squid, ils retraduisent les requêtes d'un client comme si elles etaient issues d'eux-même et, une fois la réponse obtenue, ils la recrachent au client. La plupart cache également les réponses aux requêtes pour optimiser l'utilisation de la bande passante du réseau de raccordement. Il présente en général des règles de filtrage sur adresse IP permettant de limiter l'accès à leurs services mais aussi des règles de filtrage sur le type de données qui les traversent. Dans le cas de réseaux locaux comportant un grand nombre de machines, la charge subie par le serveur peut être importante. Il convient donc de bien dimensionner la machine qui jouera ce rôle. Le paramètre essentiel est la taille de la cache mémoire qui doit être ajusté de façon à éviter au maximum la pagination. Il est également préférable d'utiliser une machine comportant au moins deux cartes réseaux pour éviter au maximum l'engorgement.
– Les proxies d'adresse,
Il s'agit de systèmes se comportant comme des routeurs IP mais qui, en plus, travestissent (masquerade in US) les paquets pour faire croire que la requête vient d'eux. Ce système induit que tout votre réseau sera percu de l'extérieur comme une seule machine et donc facilite l'établissement d'accès réservé. Le principal problème de ces protocoles est que l'initiation de la connexion doit être fait par une machine du réseau local, ce qui est un atout en terme de sécurité, mais limite les possibilités de répartition des services.

3.3 Clients WEB

Les clients Web sont principalement les navigateurs, mais aussi des systèmes tel que les robots d'aspiration des moteurs de recherche.

3.3.1 Netscape

Ce navigateur a débuté sa carrière en 1994 et a connu de nombreuses versions. La société Netscape a tenté de le vendre un certain temps puis, face à l'adversité et à une concurrence acharnée de Microsoft, il est devenu gratuit pour tout usage.

Il est compatible avec un grand nombre de sites dans ses versions récentes. Sa version communicator est conçue comme un outil intégré regroupant toutes les fonctionnalités nécessaires à l'internaute. Il est disponible pour de nombreuses architectures et permet donc un déploiement multiplateforme intéressant.

3.3.2 Internet Explorer

C'est le navigateur le plus utilisé car il est livré avec les OS Microsoft. Il en existe une version confidentielle pour Solaris Sparc et c'est le seul portage en dehors du monde Microsoft, à ma connaissance. Il est très bien interfacé avec le système Windows, ce qui facilite son utilisation par des néophytes. Son principal défaut est de ne pas intégrer l'outil de mail.

3.3.3 Autres

Dans le domaine des navigateurs WEB, il en existe plétort. Ils sont en général dédiés à une utilisation particulière : Konqueror comme gestionnaire de fichiers pour KDE, Hotjava de SUN comme browser Java, etc. Aucun n'atteind à l'heure actuelle les produits cités ci-dessus, même si leurs fonctionalités sont le plus souvent suffisantes.

Pour les robots d'exploration, importation Wget sous UNIX me semble le plus intéressant même s'il ne permet que très difficilement de créer un miroir fonctionnelle d'un site reflétant sa structure.

3.4 Contenus statiques

Dans cette partie, il s'agit de décrire le contenu statique des serveurs WEB. Il s'agit principalement de textes HTML et d'images. D'autres types de fichier peuvent être mis en ligne mais leur emploi doit être modéré pour faciliter la lecture du site à l'aide d'un simple navigateur.

3.4.1 Texte HTML

HTML est un langage de description de page basée sur la notion de balises indiquant au navigateur comment formater le document. Voici un petit exemple :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>

        <TITLE>The Gouinaud</TITLE>
</HEAD>
<BODY>
<H1 ALIGN=CENTER>Christophe GOUINAUD
</H1>
<P ALIGN=CENTER>Le Gouinaud un chercheur pr&egrave;s de chez vous ...
</P>
<P>&Ccedil;a c'&eacute;tait ma t&ecirc;te un bon jour :
</P>
<P ALIGN=CENTER><IMG SRC="moi.gif" NAME="Image1" ALIGN=BOTTOM WIDTH=166 HEIGHT=225 BORDER=0>
</P>
<UL>
<LI><P STYLE="margin-bottom: 0cm">Ma <A HREF="
http://gargamel.univ-bpclermont.fr/~gouinaud/mauj.html">t&ecirc;te
</A>un mauvais jour.

<LI><P STYLE="margin-bottom: 0cm">Mon <A HREF="
http://gargamel.univ-bpclermont.fr/~gouinaud/recher.html">sujet
                </A>de recherche.
                </P>
</UL>
<P>Pour me joindre <A HREF="mailto:gouinaud@isima.fr"><I>gouinaud@isima.fr</I></A>.
</P>
```

□

FIG. 2 – Image lisse

□

FIG. 3 – Image granuleuse

```
</BODY>
</HTML>
```

Tout informaticien un peu chevronné aura compris :

- que les balises sont encadrées par les signes $<$ et $>$.
- que les ordres de formatage commencent par une balise $< ORDRE >$ et ce termine par une balise $< /ORDRE >$.
- que les ordres de formatage peuvent être imbriqués.
- que le document commence par $< HTML >$ et finit par $< /HTML >$.
- qu'il est constitué de deux parties principales : le $< HEADER >$ et le $< BODY >$ comportant les ordres de formatage respectivement externe et interne à la page.

Il ne s'agit pas du seul langage de ce genre. Postscript ou LaTeX que j'utilise présentement en sont d'autres. Ce système de balises permet une interprétation du fichier au fur et à mesure de son arrivée, ce qui est un atout non négligeable quand le débit de téléchargement est insuffisant.

Ce document ne vise pas à expliquer l'ensemble des balises HTML. Je vous laisserai donc vous référez aux documents que l'on trouve sur le site www.w3.org.

la notion la plus importante en HTML est la notion de liens. Leur rôle est de faciliter le chaînage des actions de l'utilisateur et permet de donc de surfer de document en document. Leur syntaxe est de la forme :

```
<A HREF="URI"> Texte .... </A>
```

Le texte encadré dans la balise apparaîtra d'une façon particulière, en général souligné, pour signaler à l'utilisateur qu'il peut accéder à un nouveau document pointé par l'URL Uri. La seule difficulté est l'écriture de ceux-ci qui peuvent être soit complets (http ://garp.isima.fr/guilde/), soit absolu sans indication de hostname et de protocole (/guilde/), soit relatif (../guilde). Quelque soit la machine, ils doivent être écrits avec des / comme séparateur de répertoire et si le protocole, le host ou le répertoire sont absents, le navigateur les complétera à l'aide du contexte courant. Il ne faut donc employer les noms complets que quand c'est absolument nécessaire, lors d'un changement de site par exemple.

Les URL pointées peuvent être de nature différente. L'exemple donné au début de cette partie vous montre l'utilisation du protocole smtp.

L'autre balise remarquable est <IMG SRC="Image"> qui provoque le chargement dans le document courant de l'image désignée par la balise. Sauf configuration contraire, les browsers réaliseront automatiquement le chargement des images. Il faut donc faire en sorte qu'elles soient de taille raisonnable pour ne pas conduire à des temps de chargement de page délirants.

3.4.2 Image GIF/JPEG

Les deux formats reconnus par les broseurs sont GIF[5] et JPEG. Le gif est un format a 256 couleurs maximum, compressé en LZW par blocs. Le Jpeg est un format true color (2^24 couleurs sans problème) compressé de façon destructive.

Comme cela est dit dans la partie précédente, les images doivent contenir le moins d'octets possible. Du fait que les deux formats utilisés sur le web sont compressés, la taille n'est pas en rapport linéaire avec la dimension de l'image. Il faut donc faire très attention à la taille des fichiers quand on utilise des logiciels de traitement d'images pour rendre celles-ci plus jolies. Par exemple, l'ajout d'une texture granuleuse conduit le plus souvent à une augmentation de la taille de l'image de la figure 2 fait 25 Ko alors que l'image de la figure 3 fait 35 Ko pour une taille et un nombre de niveaux de gris présents rigoureusement identiques.

Il faut aussi prendre garde au fait que les écrans d'ordinateur ont une résolution comprise entre 72 et 100 pixels par pouce. Il est donc inutile de scanner une image à une résolution supérieure si le but est de l'afficher sur le WEB.

[5]Propriété d'AOL

3.5 Page dynamique

Les pages dynamiques sont ainsi appelées car elles s'adaptent aux requêtes de l'utilisateur. Comme rien ne se fait par miracle, ce phénomène est en général le résultat de l'exécution de programme sur le serveur ou sur le client.

3.5.1 Théorie

Deux types de scripts : ceux qui sont du côté du serveur et ceux qui sont du côté du client.

- Scripts serveur
  Ils se divisent en CGI externe au daemon et scripts exécutés grâce à un module du serveur. Si le résultat est le même, les régles de programmation seront complètement différentes.
  La deuxième catégorie de scripst est le domaine des langages ASP, PHP et des servlet Java. Il s'agit de scripts qui sont exécutés par un module interne au serveur (ie : compilés en même temps que lui) et donc ne conduisent pas au lancement d'un nouveau programme. Ces langages, ou leurs librairies, sont fortement orientés web et procurent donc des API permettant de faire face à un grand nombre de problèmes.
  Les CGI sont des programmes externes. Ils peuvent être écrits en n'importe quel langage à partir du moment où la machine qui tourne le serveur sait l'interpréter ou exécuter le binaire correspondant. Les plus utilisés sont le perl et le C, mais il est souvent plus simple d'utiliser des programmes écrits en shell sh. Bien evidemment, l'utilisation de langages compilés accélère le procédé de chargement du CGI et donc l'exécution de l'application. Il faut noter qu'il est maintenant possible de compiler un interpréteur perl comme module d'un serveur WEB. Dans ce cas, les programmes introduits dans les pages web ne seront plus des cgi mais des scripts.
  Dans le domaine du scripting, il faut éviter de réinventer la roue à chaque script, c'est-à-dire qu'il faut utiliser au maximum des choses déjà développées et non réécrire toutes les routines de gestion (par exemple : traduction d'accents) sous peine de perdre son temps et d'introduire un tas de nouveaux bugs.
  La principale caractéristique de ces applications est qu'elles fonctionnent sous forme de transaction. En fait un script ou un CGI renvoie au browser via le serveur un contenu statique. Aucune trace n'est gardée par défaut dans le serveur et donc le programmeur devra faire en sorte d'inscrire une trace dans le contenu renvoyé de façon que lors de la transaction suivante, on puisse retrouver la transaction précédente. Par exemple, si vous créez un programme d'annuaire avec un premier formulaire permettant de rentrer les donnees, puis un deuxième permettant de confirmer les données, il faudra que le script gérant la réception puisse retrouver à quelle entrée correspond cette confirmation.
  L'avantage de ce mode déconnecté est qu'un script ne tourne pas très longtemps. Par conséquent, les petites erreurs de programmation (fuite de mémoire ...) ne sont pas un problème.
- Script de navigateur
  Ils sont exécutés par le navigateur par l'interprétation d'un langage de plus haut niveau que le HTML. Il s'agit principalement de java et javascript. Pour être intéressant, ces langages doivent être présents sur un grand nombre de browsers et ne pas requérir de modifications importantes de celui-ci.
  langages si on veu pas se rendre incompréensible.
  On peut trouver des fonctionnalités similaires grâce au système Active X de Microsoft, mais il s'agit là d'un système d'installation de logiciels distants plus que d'un véritable système de scripts. Active X a pour principal défaut de nécessité l'installation de nombreux composants (DLL) sur le poste client et donc de présenter le risque d'introduire des virus ou des conflits entre bibliothèques de versions différentes.
  Le défaut de ce type de script est que l'on rencontre très fréquemment des incompatibilités entre les navigateurs dans leur interprétation. De ce fait, il est préférable de ne pas se précipiter pour utiliser les dernières fioritures des dernières versions mais plutôt de programmer *low tech* avec ce qui existait déjà il y a quelques années.

Les parties suivantes sont consacrées à la programmation de ces divers langages, à l'exception de Java qui sera vu dans un autre cours. Seront abordés les cgi, php et javascript, à cause de leur aspect multiplateforme.

3.5.2 CGI

Faire un script cgi, c'est écrire un programme qui va écrire dans sa sortie standard le code HTML de la page au lieu que celui-ci soit pris dans un fichier.

Le schéma de fonctionnement est donc dans ce cas celui-ci :

□

Il se passe les choses suivantes :

- Le client (A) envoie une demande au serveur (C)
- Le serveur (C) transmet cette demande au programme (httpd) serveur web
- Le programme serveur web lance le programme CGI, lui transmet les données issues de A
- Le programme CGI renvoie le résultat de son traitement au programme serveur WEB
- Le programme serveur WEB transmet ce résultat au client A

Dans la pratique, il se passe donc cela quand on clique sur une reférence web qui est un script (bouton envoi de formulaire par exemple http ://ComputerC.domain/date) :

- le client (A) ouvre une connection vers le serveur (C)
- A envoie à C le texte suivant :

```
GET /scripts/date HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent:  Lynx/2.2  libwww/2.14
From:  montulli@www.cc.ukans.edu
     * a blank line *
```

- (C) lance le programme **date**
- le programme **date** envoie à (C) sa réponse

```
Content-type: text/html
* blank line *
<HTML><title>Server Script</title>
<h2>The time is:</h2>
September 15, 1994 3:15 pm <p><p></HTML>
```

- (C) envoie la réponse à (A) en y ajoutant une entête (2 premières lignes) :

```
HTTP/1.0 200 OK"
Server: Apache 3.0"
MIME-Version: 1.0
Content-type: text/html
* blank line *
<title>Server Script</title>
<h2>The time is:</h2>
September 15, 1994 3:15 pm <p><p>
```

Dans le cas de l'utilisation d'un formulaire, l'envoi de A ressemblera à :

```
POST /cgi-bin/post-query HTTP/1.0
Accept: www/source
Accept: text/html
Accept: video/mpeg
Accept: image/jpeg
Accept: image/x-tiff
Accept: image/x-rgb
Accept: image/x-xbm
Accept: image/gif
Accept: application/postscript
User-Agent:  Lynx/2.2  libwww/2.14
From:  grobe@www.cc.ukans.edu
Content-type: application/x-www-form-urlencoded
Content-length: 150
     * a blank line *
nom=gouinaud
&prenom=christophe
&mail=gouinaud@isima.fr
```

et le programme cgi recevra simplement les trois dernières lignes comme si elles étaient tapées au clavier. Il devra donc les décoder et prendre les décisions qui s'imposent. Les scripts CGI peuvent être écrits en n'importe quel langage de programmation. On les écrit usuellement en C pour des raisons d'optimisation.

• Un petit formulaire en HTML

Voici le texte d'un petit formulaire html, comprenant deux champs textes : nom et prénom :

```
<HTML>
  <HEAD>
    <TITLE>Essai de formulaire</TITLE>
 </HEAD>

  <BODY bgcolor = "#fef0e0">
<FORM METHOD="POST" ACTION="http://garp.isima.fr/cgi-bin/fcgi">

<H4>Identification de la personne :</H4>

        <TABLE>
          <TR>
            <TD> Nom :<td> <INPUT TYPE="text" NAME="nom" Size=15>

            <TR><TD>Pr&eacute;nom : <td><INPUT TYPE="text"
                NAME="prenom" Size =30>
          <tr>

        </TABLE>
      </br>


<INPUT TYPE="submit" VALUE="OK"><INPUT TYPE="reset" VALUE="Effacement">
</FORM>

</HTML>
```

Il s'agit d'un texte HTML presque classique sauf en ce qui concerne les balises <FORM> dont le paramètre METHOD indique le type d'envoi à faire et l'url qui évoque le script est la valeur du paramètre ACTION.

• Lecture du formulaire en C

Le programme suivant décode les données du formulaire et les réécrit proprement dans le browser de l'utilisateur.

```
/*************************************************************
*       BUT :   verifier une clef et retourner un formulaire    *
*               prerempli a un utilisateur                      *
*       AUT :   C.G                                              *
*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <signal.h>
#include <sys/types.h>
#include <locale.h>
#include <ctype.h>
#include <fcntl.h>
#include <sys/file.h>
#include <unistd.h>

void progerr();
void trait_new();
void plustospace();
void unescape_url();
```

```
char *get_value();

#define MAXARGS 100
#define MAXENTRY 20000
#define MAXARGLEN 4096
#define MAXSTRLEN 4096

static char query_string[MAXENTRY];

int main(argc,argv)
int argc;
char *argv[];
{
        int i, len, key_ok=0;
        char *p, *method, *query, *addr, *buffer;
        char nom[MAXSTRLEN], prenom[MAXSTRLEN];

/* get all the normal environment CGI things...  */

        method = (char *) getenv("REQUEST_METHOD");
        if (method == NULL)
                progerr("Unknown method.");
        if (!strncmp(method, "POST", 4)) {
                len = atoi(getenv("CONTENT_LENGTH"));
                i = 0;
                while (len-- && i < MAXENTRY)
                        query_string[i++] = fgetc(stdin);
                query_string[i] = '\0';
        }
        else
                progerr("Unknown or invalid method.");


        addr = (char *) getenv("REMOTE_ADDR");


if (argc==1)
{
strcpy(nom,get_value("nom",query_string));
strcpy(prenom,get_value("prenom",query_string));

/* deux on envoie la sortie */

printf("Content-type: text/html\n\n");

printf("<HTML>");
printf("<BODY>\n");

printf("<FORM METHOD=\"POST\" ACTION=\"http://garp.isima.fr/cgi-bin/fcgi?VALID\">\n");

printf("<P>Bonjour Monsieur %s, </P> \n",nom);
printf("<P>Votre pr&eacute;nom est %s </P> \n",prenom);
printf("<P></P>\n");
printf("<P> &agrave; une prochaine fois \n</P>");

printf("<INPUT TYPE=\"submit\" VALUE=\"OK\"></FORM> ");

printf("<P>En cas d'erreur cliquer <A
```

```
HREF=\"http://garp.isima.fr/~gouinaud/patrice/vcthese.html\"> ici </A></P>\n");

printf("</BODY>");

printf("</HTML>");

}
else
{

printf("\n\n\n\n %s \n\n\n\n %s \n",query_string,argv[1]);

}
exit(0);
}


char *get_value(champ_in,chaine)
char *champ_in,*chaine;
/*************************************************************
*       BUT : attraper la valeur                                      *
*       PAR : champ                                             *
*       RET : la chaine alloue dynamiquement (dans le tas !)     *
*************************************************************/
{
char *value=NULL,champ[512];
int i,lchaine,lchamp;
int debut,fin;

strcpy(champ,champ_in);
strcat(champ,"=");

/* recherche de la clef */

lchaine = strlen(chaine);
lchamp = strlen(champ);

for (i=0;i<lchaine;i++)
  if (!(strncmp(&(chaine[i]),champ,lchamp)))
     { debut=i+lchamp; i=lchaine; }

/* on fixe la fin */

fin = lchaine;

for (i=debut;i<lchaine;i++)
   if (chaine[i]=='&')
     { fin = i; i=lchaine; }

value = (char *)malloc(fin-debut+1);

if (value == NULL)
   progerr("Malloc impossible dans get_value");

if (fin-debut)
  {
  strncpy(value,&(chaine[debut]),fin-debut);
  value[fin-debut]=0; /* Putain : font chier chez SUN */
                      /* si strlen(chaine)>fin-debut alors */
```

```
                    /* alors pas de zero a la fin = 1 heure*/
  }
     else
       value[0]=0;


plustospace(value);
unescape_url(value);

return(value);
}


void progerr(errstring)
     char *errstring;
{
        printf("Content-type: text/html\n\n");
        printf("<title>VPOSTE Error</title>\n");
        printf("<h1>VCPOSTE Error</h1>\n");
        printf("This program encountered an error:<p>\n");
        printf("<code>%s</code>\n<p>\n", errstring);
        exit(0);
}

void plustospace(str)
char *str;
{
    register int x;

    for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
}


char x2c(what)
char *what;
{
    register char digit;

    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
}

void unescape_url(url)
char *url; {
    register int x,y;

    for(x=0,y=0;url[y];++x,++y) {
        if((url[x] = url[y]) == '%') {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
    }
    url[x] = '\0';
}
```

Ce programme est structuré de la façon suivante :

– Initialisation
  On récupère les paramètres du CGI, la méthode utilisée, la longueur de la QUERY_STRING (chaîne des données du formulaire), on stocke la QUERY_STRING dans une chaîne de caractères c (que l'on n'oublie pas de terminer) en la lisant dans l'entrée standart.
  Un exemple de lecture de l'adresse de l'appelant est donné à toutes fins utiles.
– Récupération des données
  On saucissonne la chaine à l'aide de la fonction get_value en utilisant les noms des champs du formulaire. Afin d'éviter les ennuis, il est préférable d'utiliser une routine benaise recherchant des paterns plutôt que la fonction strtok du C. En effet, si un petit malin essaye de bricoler votre serveur, l'utilisation de pointeurs ne peut que lui faciliter la vie. Dans ce cas, les noms des champs doivent être conçus pour ne pas risquer d'être confondus avec les données [6].
  La routine de récupération des données réalise aussi le remplacement des + par des espaces (fonction plus_to_space) et le recodage des caractères spéciaux dans la table ASCII de la machine (fonction unescape_url) .
  Notez ici qu'il faut contrôler que les champs que l'on lit sont les bons et ne contiennent pas de bizarrerie.
– Ecriture de la réponse
  Une bonne réponse doit toujours commencer par une ligne Content-type, suivi de deux lignes vides, pour que le browser ne coupe pas la connection et se rende compte que le header est fini.
  Ensuite, on écrit son texte en HTML à coups de printf dans la sortie standart et on ferme la session en quittant le programme. On pourrait fermer la connection avec le serveur WEB en faisant fclose(stout) si cela s'avérait nécessaire pour faire des opérations supplémentaires sans monopoliser davantage la socket.

On constate donc qu'un CGI reçoit sur l'entrée standart les données qu'il a à traiter et répond sur la sortie standart les données qu'il veut afficher. Bien évidemment, il a tout le loisir de traiter et stocker les informations qu'il reçoit avec la seule restriction qu'il n'a que les droits du serveur WEB (nobody en général). Il faut aussi remarquer dans cet exemple qu'il charge une partie du header de la réponse et que c'est lui qui y met fin.

• Différence entre POST et GET

Nous avons vu que si nous utilisons la méthode POST, le code reçoit les données du programme dans une ligne, qui vient en fin d'entête. Si nous utilisons la méthode GET, les paramètres seront passés dans l'URL lui-même. Pour le formulaire précédent, cela donne une requête de la forme :

```
http://localhost/cgi-bin/fcgi?nom=gouinaud&prenom=h%E9l%E9ne
```

Dans ce cas le programme reçoit les données du formulaire dans ses arguments (argv[1], argv[2], ... en C par exemple). Remarquons ici le codage des lettres accentuées et des espaces qui est le même que précédemment.

Nous avons la possibilité de passer des arguments supplémentaires à un programme CGI uniquement dans le cas où le formulaire POST les données. Dans ce cas, il suffit d'écrire les arguments optionnels dans l'URL désigné dans la balise déclarant le formulaire par exemple :

small

```
<FORM METHOD="POST" ACTION="http://localhost/cgi-bin/fcgi?pouet=3&tut=2">
```

Les arguments sont écrits avec un ? pour le premier puis des & pour les suivants. Ces arguments permettent de rendre multi-usage un script CGI en fournissant des informations supplémentaires. Nous devons ensuite dans le script récupérer les valeurs de ces arguments de façon à aiguiller correctement le traitement.

L'utilisation de la méthode GET sera plutôt (comme le prévoit la norme) réservée à faire des requêtes sur des bases de données et la méthode POST + paramètres permettra de faire une gestion complète des formulaires complexes.

• Suivi des sessions

Un autre problème de la programmation des CGI est posé par l'aspect déconnecté des transactions HTTP. Comme nous l'avons vu plus haut, la connection entre le client et le serveur est fermée après un unique aller et retour. Dans le cas où l'on veut confirmer les données que l'utilisateur a envoyé, ou utiliser plusieurs formulaires pour permettre une

[6] ce n'est pas le cas ici, M. prenom ne pourra jamais utiliser ce service.

sélection des questions, les formulaires successifs renvoyés par le script devront comporter des données permettant de pister l'utilisateur.

il esiste pour cela deux méthodes :

– Les variables cachées
  Cela consiste à cacher dans la page renvoyée les informations dont nous pouvons avoir besoin par la suite. Cela peut se faire soit par ajout de champs préremplis au formulaire, mais écrit dans la couleur de fond de la page, soit par l'ajout de paramètres et un emploi d'une méthode POST.
  Par exemple, si on voulait garder les données du formulaire pour les passer à un suivant, on écrirait :

  ```
  <FORM METHOD="POST" ACTION="http://localhost/cgi-bin/fcgi?nom=gouinaud&prenom=christophe">
  ```

  C'est de cette façon que sont en général gérer les pages de réponses multiples des moteurs de recherche. Le défaut de cette méthode est la complexité des champs à mettre en oeuvre et la fragilité induite par le codage-décodage multiple. Son avantage principal est qu'elle permet de ne pas garder de trace temporaire des données et donc évite de devoir gérer les sessions qui n'ont pas abouti. Une autre technique est d'utiliser des champs invisible dans les formulaire de type INPUT TYPE HIDEN.
– Les identifiants de session
  Il s'agit dans ce cas de stocker les paramètres de la session, c'est-à-dire les données des formulaires sur le serveur, et de renvoyer au client un numéro de référence identifiant ces données. Nous pouvons faire cela par l'envoi d'une variable cachée ou d'un cookie. La deuxième méthode doit être implémentée à l'aide d'une librairie. Une bonne description des cookies peut être trouvée à http ://developer.netscape.com/docs/manuals/communicator/jsguide4/cookie.htm.

Dans tous les cas, si l'on souhaite faire confirmer à un utilisateur des données, il est souhaitable de lui envoyer un formulaire prérempli permettant la conformation et la modification.

Pour faire cela, le plus simple est d'écrire le formulaire à renvoyer dans un fichier html en remplaçant le contenu des champs par un code facile à repérer. L'envoi du formulaire prérempli se fera par envoi de ce fichier dans lequel on remplacera les codes par leur valeur à la volée. Un exemple de code possible est :

```
<TEXTAREA name="Profil" ROWS=6 COLS=50 >CGXX_Profil</textarea>
```

Le code CGXX permet de repérer qu'il s'agit d'une variable et donc d'insérer le contenu du champ à la volée.

• Environnement des scripts CGI

Outre les données d'un formulaire, un script cgi reçoit un certain nombre d'informations par des variables d'environnement. Celles-ci sont en fait héritées du processus père (httpd) et peuvent être utiles.

```
SERVER_PORT=80
GATEWAY_INTERFACE=CGI/1.1
PWD=/home/httpd/cgi-bin
HTTP_USER_AGENT=Mozilla/4.6[en](X11;I;Linux 2.2.9-19mdk i686)
HTTP_REFERER=http://elo.isima.fr/~gouinaud/patrice/vcthese.html
REMOTE_PORT=1202
HTTP_HOST=localhost
SCRIPT_FILENAME=/home/httpd/cgi-bin/fcgi
HTTP_CONNECTION=Keep-Alive
HTTP_ACCEPT_ENCODING=gzip
HOSTNAME=elo.isima.fr
SERVER_SIGNATURE=
SERVER_ADMIN=root@localhost
REQUEST_METHOD=POST
MACHTYPE=i586-pc-linux-gnu
SCRIPT_NAME=/cgi-bin/fcgi
REMOTE_ADDR=127.0.0.1
SERVER_SOFTWARE=Apache/1.3.6 (Unix) (Mandrake/Linux) PHP/3.0.8
CONTENT_LENGTH=13
QUERY_STRING=pouet=3
```

```
SERVER_NAME=elo.isima.fr
SHLVL=1
SERVER_PROTOCOL=HTTP/1.0
HTTP_ACCEPT=image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,image/png,*/*
REQUEST_URI=/cgi-bin/fcgi?pouet=3
DOCUMENT_ROOT=/home/garp/web/
HTTP_ACCEPT_LANGUAGE=fr,en
SHELL=/bin/sh
HOSTTYPE=i586
OSTYPE=linux-gnu
TERM=dumb
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin
HTTP_ACCEPT_CHARSET=iso-8859-1,*,utf-8
CONTENT_TYPE=application/x-www-form-urlencoded
```

Le programmeur fera son marché là-dedans. Les informations intéressantes étant bien évidemment celles qui concernent le browser utilisateur du service. Remarquons aussi les chaînes comme PWD qui permettent de savoir le répertoire de base de l'exécution du script.

Cette liste est obtenue en utilisant un simple programme en shell contenant la commande env.

• Utilisation d'autre langage

Comme nous l'avons vu plus haut, un script CGI prend ces données dans l'entrée standart et les recrache sur la sortie standart. N'importe quel langage peut donc être utilisé pour programmer ce type d'application. Si le fortran 77 présente peu d'intérêt dans ce cadre, l'utilisation de langage interprété peut faciliter grandement la mise au point des programmes.

La grande majorité des CGI sont écrits en perl, qui est un langage simple, compilé à la volée, doté d'une énorme librairie permettant d'accéder à beaucoup d'API. Sa syntaxe est proche de celle du langage C et des commandes de vi. En perl, il faut toujours aller jeter un coup d'oeil a http ://www.cpan.org/ avant de développer quelque chose car la majorité des choses y sont déjà faites (et bien).

Pour des petits scripts, on peut aussi utiliser le shell de Bourne et les commandes UNIX classiques qui permettent énormément de manips. A titre d'exemple, voici le traitement du formulaire précédent en shell sh :

```
#!/bin/sh

TMP=/tmp/essai.$$

#prise de contact avec le browseur

echo 'Content-type:  text/html'
      echo ''
      echo ''
      echo '<HTML>'

#lecture du formulaire
read FORMULE
#on vire les &
LISTE=‘echo $FORMULE | tr \& ' ' ‘

#on vire les + et on ecrit ligne à ligne dans le fichier courant
# ca fera des lignes de la forme nom='gouinaud'
#
for i in $LISTE
do
echo $i\' | tr '+' ' ' | sed s/=/=\'/ >> $TMP
done
```

```
#Kolossale finesse, on recharge les variables
#ainsi dans $nom -> gouinaud
. $TMP

#sauvegarde des resultats

echo $nom';'$prenom';' >> /tmp/sauvegarde
echo '---'  >> /tmp/sauvegarde

echo 'SUBJECT: Inscriptions de' $nom ' '$prenom > $TMP
echo ' ' >> $TMP
echo 'Merci d'\''avance' >> $TMP

EMAIL='postmaster'

/usr/sbin/sendmail -f $EMAIL root < $TMP

#affichage des resultats dans le broseur
echo 'Resultat des courses <B> TOTO </B>'
echo '<P>' $FORMULE '</P>'
echo '<P>' $LISTE   '</P>'
echo '<P> Le prenom entre est' $prenom 'et le nom est' $nom '</P>'
echo '</HTML>'

#nettoyage
if [ -f $TMP ]; then
 /bin/rm $TMP
fi
```

• Organisation et conclusions

Pour faire une application cgi rapidement, il convient de procéder comme suit :

- Commencer par décrire l'application sous forme d'un graphe papier
- Produire toutes les pages WEB statiques de l'appli
- Produire tous les formulaires et les sous formulaires
- Soumettre les formulaires aux décideurs
- Programmer le CGI

De cette façon, vous perdrez un minimum de temps par des erreurs de conception. Pour debugger, utilisez des fichiers de trace. Il est aussi possible de lancer le programme en ligne de commande ou sous debugger en lui passant la query string à la main et en désactivant les lignes de contrôle de la méthode d'envoi. Il faut toujours prévoir une vérification des informations sensibles. Par exemple, une adresse email ne doit contenir qu'un @, pas de ; et pas d'espace. Ignorer cette règle revient à créer un trou de sécurité majeur.

Par exemple nobody@isima.fr ; cat /etc/passwd | mail gouinaud@isima.fr permet de récupérer le fichier passwd de la machine si votre cgi est mal tarponné.

Il faut toujours prendre soin de l'utilisateur final. Pensez qu'il est à l'autre bout du monde et faites en sorte de lui mettre devant les yeux toutes les informations nécessaires à un instant donné plutôt que le canard à Mimile en 3D.

3.5.3 PHP

Le php est un langage de script conçu spécifiquement pour la production de pages WEB. Il est interprété par un programme qui tourne côté serveur et la plupart du temps dans un module du serveur.

Il s'inspire de trois sources linguistiques :

- le shell, duquel il hérite sa commande echo et ces notions de chaînes de caractères

– le perl dont il hérite une partie des structures de données et en particulier les tableaux associatifs
– le C dont il hérite une grande partie de ces opérateurs, de sa librairie de fonctions, de ses structures de contrôle et de sa méthode pour transmettre les paramètres à une fonction.

Les scripts php sont en général inclus dans du code html de la façon suivante :

```
<HTML>

<TITLE> Test en php </TITLE>

<? /* debut du script */

phpinfo();

?> /* fin du script */

<P>Patience pour la dexieme partie </P>

<? $date=date("d/M/Y"); ?>

<P>On va donne la date </P>

<? echo "aujourd'hui on est le $date"; ?>

</HTML>
```

Comme on le voit, les instructions php sont séparées du code HTML par les symboles < ? et ?>. Quelque soit le nombre de débuts et de fins, l'ensemble du code php de la page est considéré comme un seul script. Vous en déduirez aisément que les variables auront par défaut la portée du script et donc d'une transaction http.

• syntaxe de base

Le php s'inspire du C, donc :

– Les instructions se terminent par un ;
– Les blocs d'instruction commencent par { et finissent par }
– Les variables commencent par un $ et leurs noms sont composés de lettre, chiffre et underscore. Elles n'ont pas besoin d'être déclarées et, dans ce cas, ont auto-typé lors de la première affectation.

```
$x=1.0;
$n=1;
$nom="gouinaud";
echo "$nom $x $n";
```

Les types de données sont principalement entier, réel et chaîne, appelés respectivement int, double et string. Php4 introduit booléen et ressource.
Bien évidemment, les types primaires peuvent être étendus en tableau (array) et en objet[7] (object).
– Les fonctions ont en gros la même syntaxe qu'en C ANSI, si ce n'est que le type n'a pas besoin de figurer dans l'entête. Elles sont déclarées par le mot clef function. Les paramètres sont passés par défaut par valeur. Les passages par référence (ie : par variable) se font en préfixant la déclaration du paramètre du signe & comme dans l'exemple suivant :

```
function point_virgule(&$chaine)
{
   $chaine=ereg_replace(".",",",chaine);
}

function affiche_nombre($chaine,$money="Brouzouf")
```

[7] php4 uniquement

```
{
   echo "Vous me devez $nombre brouzoufs";
}

$nombre="1000.00";
point_virgule($nombre);

affiche_nombre($nombre);
```

On peut définir des argument optionnels en ajoutant =valeur à la déclaration du paramètre comme cela est fait dans l'exemple ci-dessus pour la fontion affiche_nombre.
Bien évidemment, les variables ont par défaut la portée de la fonction sauf si elles ont déja été employées en dehors de toute fonction. On peut globaliser une variable en utilisant le mot-clef global. On pourra y accéder via le tableau associatif $GLOBALS.
Les fonctions doivent être déclarées et interprétées avant d'être utilisées. Leur corps doit donc être écrit avant leur utilisation. Notez bien que ceci est différent du langage C, qu'il n'existe pas de possibilité de prédéclaration façon prototype ou forward (Pascal), ce qui supprime les risques d'étreinte fatale.
Il est de bon goût de regrouper les fonctions utilisées par tout le script d'une application dans une librairie. Dans la pratique, il suffit d'écrire toutes les fonctions dans un fichier et ensuite de les charger dans le script à l'aide de la fonction require ou include.

```
require("lib.php");
include("lib2.php");
```

Moyennant tout cela, il y a moyen de structurer un code de façon à ce qu'il ne ressemble pas à un intestin de cochon reconditionné.

- Les opérateurs sont les suivants :
  - Arithmétiques :
    - $(+, -, *, /, .)$: addition, soustraction, multiplication et division, concaténation
      ex : $a+$b, "chien"."chat",
    - $(++, --)$: incrémentation, décrémentation
      $a++ est équivalent à $a=$a+1.
  - De comparaisons :
    - $(<, >)$: plus petit, plus grand
    - $(<=, >=)$: de même mais avec égal
    - $(==, !=)$: égal et différent
  - logique
    - (&&,AND) : **et** logique
    - (||,OR,XOR) : **ou** logique
  - D'affectation :
    - $(=)$ affectation de valeurs **(A ne pas confondre avec le == de la comparaison)**
    - $(+=, -=, *=, /=, .=)$: addition puis affectation, etc, ...
      En fait $i/=4$ est identique à $i = i/4$ mais en plus rapide !
  - de conversions
    - ((string)) : conversion en chaîne
    - ((int)) : conversion en entier. synonyme : (integer)
    - ((real)) : conversion en double. synonymes : (double),(float)
    - ((array)) : conversion en tableau
    - ((object)) : conversion en objet

  Les expressions sont des suites de variables séparées par des opérateurs. Elles sont évaluées de gauche à droite avec les règles de priorité habituelles.

• Variable, tableau et tableau associatif

La déclaration de tableau se fait sans spécifier la taille.

```
$machine= array();
```

On leur affecte des valeurs comme suit :

```
$machine[0]= "pomme";
$machine[1]= "banane";
$machine[2]= "orange";
```

ou au moment de la déclaration :

```
$machine= array( "pomme", "banane", "orange" );
```

Voici quelques fonctions permettant de manipuler les valeurs d'un tableau

- (sizeof($tab)), nombre d'éléments d'un tableau. équivalent de count
- (is_array($tab)), est-ce un tableau ?
- (reset($tab)), place le compteur sur le premier élément du tableau : chaque variable tableau possède un compteur repérant l'élément courant qui permet un parcours similaire à une liste de pointeurs.
- (end($tab)), place le compteur sur le dernier élément.
- (current($tba)), retourne la valeur courante associée au compteur.
- (next($tab),prev($tab)), manipule le compteur en avant et en arrière. Ces deux fonctions renvoient FALSE si l'opération est impossible.
- ($comp=each($tab)) renvoit l'index et la valeur courante correspondante dans un tableau à 2 elements ; $comp[0] contient le compteur et $comp[1] la valeur.
- (key($tab)) renvoit l'index de l'élément courant du tableau.
- ($tabt=sort($tab)), fonctions de tri de tableau. Il en existe de nombreuses.

Le php reprend la notion de tableau associatif, c'est à dire indexé par des chaînes de caractère et non par des entiers. Comme l'index est une chaîne de caractères, le système d'indexation repose sur des tables de hachage et donc on les appelle aussi "hash array" ou "hash". Il se déclare comme un tableau traditionnel. La distinction se fera lors de l'affectation.

```
$ip["pommes"]= "192.168.100.1";
$ip["banane"]= "192.168.100.2";
$ip["litchie"]= "192.168.100.3";
```

Les fonctions spécifiques à ces tableaux permettent de tenir compte de ces index un peu particuliers :

- (isset) ,tester l'existence d'un élément,
  ```
  if( isset( $ip["kiwi"] ) ) {
          ping($ip["kiwi");
  }else{
          echo "Unknow Host\n";
  }
  ```
- asort, arsort, ksort, aksort, fonctions de tri, asort trie par valeurs croissantes, arsort par valeurs décroissantes ,ksort trie par index (key) croissantes

Les tableaux associatifs sont un des intérêts majeurs du PHP pour la programmation de pages WEB, car il permet l'association de champs de formulaire à des noms de variable.

Les tableaux multidimensionnels se déclarent par appel imbriqué de la fonction array :

```
double $matrice=array(array());
```

WAR : Attention à l'occupation mémoire des tableauc de dimension supérieure à deux.

• Structure de contrôle

Les structures de contrôle sont classiques et différent du C principalement par l'introduction du **elseif**.

- for : La boucle **for** permet de répéter un bloc d'instruction conditionnellement à un test en exécutant une instruction d'incrémentation à chaque fois. La boucle exécutera "instruction 2" tant que le test de boucle est vrai. L'instruction d'incrémentation sert à modifier les variables intervenant dans la boucle. Sa syntaxe est de la forme :

for (instruction 1 ; test de boucle ; instruction d'incrémentation)
{instruction 2 ;}

```
for ($i=0;$i<100;$i++)
  echo $i;
```

Le schéma d'exécution de l'instruction **for** est :

– Premier passage
  L'instruction 1 est exécutée, le test est évalué. Si il est vrai, l'instruction 2 est exécutée, sinon, on quitte la boucle.
– Passage suivant
  L'instruction d'incrémentation est exécutée, le test est évalué. Si il est vrai, l'instruction 2 est exécutée, sinon, on quitte la boucle.

Dans le cas où la boucle n'est constituée que d'une seule instruction, les accolades ne sont pas nécessaires.

– Boucle *while*

La boucle **while** répète un bloc d'instructions tant qu'une condition est vraie. Elle a deux formes syntaxiques :

while (condition) {instructions ;}

ou :

do {instructions ;} while (condition) ;

La condition est exprimée sous la forme d'une expression (cf 2.3). Dans le premier cas, la condition est testée avant d'entrer dans la boucle, et si celle-ci est vraie, alors le programme peut faire les instructions. Dans le deuxième cas, le programme rentrera automatiquement dans la boucle et il vérifie ensuite la condition. Pour résumer, dans le premier cas, si la condition est fausse dès le départ, les instructions ne sont jamais faites, alors que dans le deuxième cas, on passe obligatoirement une fois dans la boucle.

– Attention, ne pas oublier de faire varier à l'intérieur de la boucle les variables constituants la condition (car autrement vous resterez éternellement dans la boucle).
– Attention, toutes les variables dans une condition doivent avoir obligatoirement une valeur avant d'être testées.

Exemples :

```
$i=1;
while ($i<2)
 {
   echo $i;
   $i++;
 }
```

Si on affecte 3 à i, alors la boucle ne sera jamais exécutée. Par contre, dans le cas suivant, elle le sera :

```
$i=1;
do
 {
   echo $i;
   i++;
 }
while ($i<2);
```

Tout dépend donc de l'effet recherché ...

Il faut remarquer que : while (expression) est équivalent à for (;expression ;) mais il vaut mieux n ;utiliser le "for" qu'à bon escient si on ne veut pas rendre illisible ses programmes.

– Test *if*

Les instructions **if** et **else** permettent d'exécuter des instructions conditionnellement à une expression. Ils servent donc d'instruction de branchement. Il n'y a pas de then.

if (expression) {instructions 1 ;}
[else {instructions 2 ;}]

Le **else** est facultatif mais bien pratique dans le cas où on veut exprimer un "sinon". Si l'expression est vraie, ce sont les instructions 1 qui sont effectuées, sinon ce sont les expressions 2 à condition qu'elles existent. La condition est toujours placée entre parenthèse (et il n'y a pas de ; après). Attention à ne pas confondre la comparaison "==", et l'affectation "=" (faute malheureusement courante et difficile à détecter).

Exemple :

```
if ($i==3)
    print("i est egale a trois\n");
  else
    {
      print("i n'est pas egale a trois\n");
      print("mais cela ne saurait tarder ..\n");
      $i=3;
    }
```

Il est possible de mettre un *if* à l'intérieur d'un autre *if* et d'utiliser **elseif** pour faciliter la lecture.

– switch

Il permet de d'aiguiller un traitement suivant une variable. Il rend un code plus efficace et plus lisible qu'un empilement de tests if.

```
        switch( [variable] ) {
                case [valeur1] :
                        [bloc d'instructions]
                        break;

                case [valeur2] :
                        [bloc d'instructions]
                        break;
                ...
                default:
                        [bloc d'instructions]
        }
```

La valeur de la variable est comparée successivement à chaque case. Si il y a égalité, le bloc d'instructions est exécuté. Il ne faut pas omettre le break en fin de bloc, sans quoi toutes les instructions jusqu'au prochain break sont exécutées[8].

La constante default permet de définir des instructions à effectuer par défaut, si aucun cas ne correspond.

```
switch( $prenom ) {
case "bob" :
case "toto" :
case "julien" :
        echo "bonjour ", $prenom , " !  vous etes un garçon";
        break;
case "anne":
case "béatrice" :
case "patricia" :
        echo "bonjour ", $prenom , " !  vous etes une fille";
default:
        echo "Bonjour $prenom ! Désolé je ne sais pas !"
}
```

• Entrees sorties

Ce langage est conçu principalement pour réaliser des entrées sorties via des pages WEB ou des fichiers. Comme il s'agit ici principalement de notre objet nous nous limiterons à l'étude des fonctions utiles dans ce domaine.

– Instruction echo et function print

La fonction echo affiche un (ou plus) argument. Si l'argument est une chaîne entre simples quotes ' il est affiché tel quel.

```
echo 'Hello, you';
```

Avec le quote double " les variables contenues dans cette chaîne sont interprétées.

```
$nom= "Toto";
echo "Hello, $nom"; // Hello, Toto
```

[8] En fait, comme en C, le case valeur : n'est qu'une étiquette et ne donne pas lieu à la génération d'instructions machines

```
echo 'Hello, $nom'; // Hello, $nom
```
On peut également inclure le résultat d'une fonction directement dans un echo.
```
        echo "Votre Nom en majuscule : ", strtoupper( "Toto" ), "\n";
```
Pour afficher le caractère “, on l'escape à l'aide du caractère d'échappement
```
echo " Escaping de caractères : \" \n";
```
On peut inclure des caractères spéciaux pour contrôler le flux affiché :
- n̂ saut de ligne
- r̂ end of line
- t̂ tabulation

A ce niveau, il existe une petite subtilité sur les chaînes de caractères qui est la différence entre les "chaines" et les 'chaines'
```
$toto = "gouinaud";
echo "ton nom est $toto" \\ affiche => ton nom est gouinaud
echo 'ton nom est $toto' \\ affiche => ton nom est $toto
```
En fait, dans les chaînes entre guillemets, les variables sont évaluées, alors que dans les chaines entre apostrophes (ou accent aigus), le caractère $ n'a pas de signification particulière si ce n'est d'afficher effectivement un dollar. C'est le même comportement que le shell de Bourne.
La fonction print prend un seul argument qui est la chaîne à écrire, on utilise plus souvent echo.

– Entrée sortie dans les fichiers
Pour les fichiers non structurés, le nom des fonctions est très inspiré du C.
Voici un exemple qui en dit long :
```
$fp=fopen("wp.html","r");
while (!feof($fp))
 { $ligne=fgets($fp,1024);
   print("$ligne\n");
 }
fclose($fp);
```
Les autres fonctions sont identiques au C.

– Gestion des cookies
Les cookies sont de courts textes que l'on peut demander au navigateur client de stocker. Ils sont associés à un domaine et ont une durée de vie limitable.
La création d'un cookie se fait à l'aide de la fonction setCookie(chaine nom,chaine valeur,entier expire,chaine chemin,chaine domaine, entier sécurité) ; Un cookie pour ISIMA, valable 24 heures, envoyé de façon non sécurisé, a la tête suivante :
```
setCookie("ISIMA","CLF-PASS",time()+3600,"/intranet/","www.isima.fr",0);
```
Quand le navigateur fait une requête, il compare les domaines et chemins à ceux des cookies qu'il a en stock et les postes avec sa requête si il trouve une correspondance. Le script réceptionnaire se trouve donc agrémenté d'un variable au nom du cookie $ISIMA dans le cas présent.
Cela permet d'identifier quelqu'un dans la mesure où il accepte les cookies.

• Gestion des formulaires

Le principe de gestion des formulaires est extrêmement simple. Quelque soit la méthode employée par le programmeur (POST ou GET), les champs des formulaires sont renvoyés dans des variables correspondant à leurs noms et une variable au nom bouton qui a déclenché l'action contiendra la *VALUE* du bouton. Soit le formulaire suivant :

```
<FORM ACTION="livre.php3" METHOD="post">
<BR> Titre:  <INPUT TYPE="text" NAME="Titre" VALUE="" > </BR>
<BR> Auteurs  <INPUT TYPE="text" NAME="Auteur" VALUE=""> </BR>
<BR> <INPUT TYPE="submit" NAME="FORMULAIRE" VALUE="OK" > </BR>
</FORM>
```

Dans le script livre.php3, l'interpréteur php précréera une variable $Titre, une variable $Auteur et une variable $FORMULAIRE qui vaudra "OK". Le script de traitement de la requête s'écrira donc :

```
<HTML>
```

```
<?
if ($FORMULAIRE=="OK")
  {
    echo "<P>Vous recherchez le livre $Titre &eacute;crit par $Auteur<P>\n";
  echo "<P>Il n'est pas dans la bibliothèque</P>";
  ?>
  <P><P>Vous pouvez modifier votre recherche ci-dessous<\P>
  <?
  }
  else
  {
  ?>
  <P><P>Remplissez le formulaire ci-dessous pour rechercher votre ouvrage :</P>
  <?
  }
?>
<FORM ACTION="livre.php3" METHOD="post">
<?
echo '<BR> Titre:  <INPUT TYPE="text" NAME="Titre" VALUE="'.$Titre.'" > </BR>';
echo '<BR> Auteurs  <INPUT TYPE="text" NAME="Auteur" VALUE="'.$Auteur.'"> </BR>';
?>
<BR> <INPUT TYPE="submit" NAME="FORMULAIRE" VALUE="OK" > </BR>
</FORM>
</HTML>
```

A la fin du script, on remet le formulaire que l'on prérempli en ré-utilisant les variables. Dans ce domaine, le php est extrêmement pratique par le fait que les variables non utilisées sont déclarées vides et que par conséquent, ce script peut s'autoappelé.

Nous pouvons nous servir des paramètres du script pour faire passer des variables d'état qui permettent de suivre une session facilement :

```
<HTML>
<?
if ($FORMULAIRE=="OK")
  {
    if (($EAuteur==$EAuteur) AND ($ETitre==$Titre))
    {
    echo "<P> Le monsieur te dit qu'il ne l'a pas, alors n'insiste pas !";
    exit();
    }
    else
    {
    echo "<P>Vous recherchez le livre $Titre &eacute;crit par $Auteur<P>\n";
  echo "<P>Il n'est pas dans la bibliothèque</P>";
  ?>
  <P><P>Vous pouvez modifier votre recherche ci-dessous<\P>
  <?
     <FORM ACTION="livre.php3?Etitre=$Titre&EAuteur=$Auteur" METHOD="post">
  }
  else
  {
  ?>
  <P><P>Remplissez le formulaire ci-dessous pour rechercher votre ouvrage :</P>
  <?
```

```
    <FORM ACTION="livre.php3" METHOD="post">
  }
?>

<?
echo '<BR> Titre:  <INPUT TYPE="text" NAME="Titre" VALUE="'.$Titre.'" > </BR>';
echo '<BR> Auteurs  <INPUT TYPE="text" NAME="Auteur" VALUE="'.$Auteur.'"> </BR>';
?>
<BR> <INPUT TYPE="submit" NAME="FORMULAIRE" VALUE="OK" > </BR>
</FORM>
</HTML>
```

Ce script gère trois cas : l'expédition initiale du formulaire, la réponse à un premier envoi et au suivant et la fin de session pour les pénibles. Nous constatons donc que le php permet de regrouper facilement tous les éléments d'une transaction sous un seul fichier, ce qui est un atout pour une programmation propre.

Cette gestion artisanale évite d'avoir recours aux cookies et donc de s'affranchir des paranoiaques[9] qui les refusent.

• Description de quelques fonctions

– Les fonctions sur les chaînes de caractères sont préfixées par str. Elles sont très proches du C à ceci près que les opérateurs d'égalité et de concaténation en rendent un grand nombre inutiles.

strlen(chaine)	longueur de la chaîne
chop(chaine)	suppression des espaces à la fin
trim(chaine)	suppression des espaces aux extrémités
htmlEntities(chaine)	convertit en chtmeuleu les caractères spéciaux
strToLower(chaine)	passe en minuscule
strToUpper(chaine)	passe en majuscule
strCmp(chaine1,chaine2)	comparaison de deux chaînes
strcCaseCmp(chaine1,chaine2)	comparaison de deux chaînes en ignorant la casse
strPos(chaine1,chaine2, debut)	position de chaine2 dans chaine1
strtok(chaine,character)	renvoie un pointeur sur un token

La fonction strTok s'emploie comme suit :

```
$chaine="gouinaud:501:100:Christophe Gouinaud:/home/gouinaud:/bin/tcsh"
for ($tok= strTok($chaine,":"); $tok!=""; $tok=strTok(":"))
  echo $tok;
```

ce qui est particulièrement utile pour la lecture des fichiers textes qui exportent des données de tableurs.

– Les fonctions sur les fichiers sont multiples. Elles sont identiques au langage C sous UNIX et implémentent la plupart des primitives système (sat, link, unlik).

Ce langage étant très orienté fichiers texte, la principale innovation est la fonction file qui range par ligne le contenu d'un fichier texte dans un tableau. Le script suivant affiche le contenu complet d'un fichier.

```
<?
$tabfile=file("annuaire.dat");
for ($i=0;$i<count($tabfile);$i++) echo($tabfile[$i]);
?>
```

Cette fonctionalité est aussi implémentée sous la forme d'une fonction fPassThru qui envoie le fichier dans la sortie standart. La différence de cette fonction avec include est que le fichier n'est pas interprété comme un script php, ce qui la rend plus rapide.

Php fournit aussi un support pour lire et écrire les fichiers compressés avec les fonctions préfixées par gz. Presque toutes les fonctions classiques sont implémentées :

[9] Même si le flicage qu'ils permettent est un risque potentiel pour la liberté individuelle

gzOpen(chaine filename, chaine mode, boolean include_path)	ouverture
gzPuts(integer file, string chaine)	écriture d'une chaîne
gzGets(integer file, string chaine)	lecture d'une chaîne
gzEof(integer file)	test de fin de fichier
gzSeek(integer file,integer offset)	déplacement relatif
gzClose(integer file)	allez, on ferme

Une dernière fonction utile est tmpfile() qui renvoie un entier descripteur de fichier qui désigne un fichier créé dans la zone temporaire de la machine qui sera automatiquement détruit lors de sa fermeture.

– les fonctions de date

La fonction la plus utile est date(format,timestamp) qui renvoie une chaine et qui, employé avec un seul paramètre, renvoie la date courante à un format donné.

```
echo date("Y/M/d"); => 2002/03/06
echo date("D d M y H:i"); => 2002 03 06 3 27
```

Il faut aussi citer la fonction getdate(timestamp) qui renvoie un tableau associatif avec chaque élément de l'heure, time() qui renvoie le timestamp courant et mktime(heure,minute,seconde,mois,jour,annee) qui permet de fabriquer un timestamp de son choix. Pour comparer les dates, on utilise les timetamp qui sont des entiers.

```
<?
$cur=time();
$old=mkitme(0,0,0,22,02,1966);
if ($cur>$old) {echo "on est en bcg");}
?>
```

Ainsi on peut facilement vérifier la date des infos de façon à ne pas trop laisser moisir les serveurs WEB.

Beaucoup d'autres fonctions du même genre inspirées de la librairie C existent. Le programmeur php doit se procurer un bon bouquin pour éviter de reinventer le char à banc toutes les cinq minutes.

• Conclusions

Le php est un langage particulièrement adapté à la programmation des pages WEB.

3.5.4 Javascript

Il s'agit d'un langage de script mais interprété par le browser. Ceci induit quelques limitations :

– La compatibilté du code n'est pas garantie. Il faut donc toujours tester son code complètement avec la plupart des navigateurs courants.

– Ces scripts tournent une fois que la transaction est complète. Il n'est donc pas possible d'enrichir les données d'une page à partir du serveur dynamiquement. Un script en javascript devra donc être conçu pour être complètement autonome.

Le but de cette partie est d'introduire les concepts de base de ce langage et d'en montrer quelques points forts.

Javascript est nécessaire dès que l'on souhaite provoquer des modifications dynamiques de l'aspect d'un document une fois qu'il a été chargé. Par exemple, on pourra modifier l'aspect d'une liste déroulante en grisant les options inutiles en fonction des choix qui sont réalisés dans un formulaire ou d'un calcul un peu lourd que l'on souhaite faire réaliser à distance.

• Un petit exemple

```
<HTML>
<TITLE> Test javascriptos ! </TITLE>
<BODY>
blalbla bla ...

<SCRIPT LANGUAGE="javascript">
cur = new Date();
document.write("Dernière modification :"+document.lastModified);
document.write("L'heure courante est :"+cur.getHours()+"h"+cur.getMinutes());
```

```
</SCRIPT>
</HTML>
```

Nous remarquons que pour intégrer un script dans une page HTML, vous devez utiliser le tag <script> comme suit :

```
<script language="Javascript">
the source ..
</script>
```

Si, par exemple, le script est dans un fichier moyenne.js, vous pouvez l'appeler comme suit :

```
<script language="Javascript" src="moyenne.js"></script>
```

En fait, le navigateur insérera à ce niveau le script contenu dans le fichier moyenne.js.

Dans cet exemple, nous remarquons que le document courant est désigné comme un objet que la méthode write permet de modifier.

• Syntaxe

La syntaxe est quasi conforme au C++ avec quelques aménagements pour faciliter l'interprétation.

- Expression ou instruction de terminant par un point virgule
- Variables de nom quelconque comportant des lettres ou des chiffres
- Types de donnée
  Principalement : entier, flottant et chaîne de caractères déterminé au moment de la première utilisation. Type booleen avec false et true.
  Structuration de données sous forme de tableaux ou d'objets.
  soit de façon déclarative : nomVariable = new Array(5), soit par initialisation : animaux = new Array(" chart ", " chien ", " cheval ", " souris ") ;
  Les tableaux à plusieurs dimensions s'initialisent de la manière indirecte suivante :
  ```
  toto = new Array(3) ;
  toto[0] = new Array(3) ;
  toto[1] = new Array(2) ;
  toto[2] = new Array(3) ;
  ```
  Nous verrons la gestion des objets plus loin, leur emploi étant par ailleurs conforme au C++. Ils servent également de structure.
- Opérateurs
  Comme en C, mais en plus pauvre. L'affectation par le = est étendue aux objets donc aux chaînes de caractères.
- Structures de contrôle
  Elles sont conformes au C, à ceci près que les tests sont de type booléen et que les adresses des cases peuvent être des chaînes de caractères.
- Fonction
  Elle se déclare comme en php, avec le mot clef function :
  ```
  function df(x)
  {
  y=2*x;
  return(y);
  }
  ```
  Le passage des paramètres se fait par valeurs par défaut. Je n'ai pas trouvé d'autre moyen que d'utiliser des objets pour les passer par références. Les fonctions doivent être déclarées et accessibles dans le code du script. Une variable déclarée à l'intérieur d'une fonction n'est visible que dans la fonction. Au contraire, si le mot clé var est omis, une fois la fonction appelée, la variable est aussi visible à l'extérieur du contexte de la fonction.

• Décortiquons la classe string

L'instance d'une classe est créée par l'intermédiaire de l'instruction new ce qui se traduit pour les strings par :

```
Logique = new String( " 2 + 2 = 4 ") ;
vide = new String() ;
```

La classe String possède certaines méthodes (fonctions) permettant de manipuler les chaînes de caractères. En voici les plus importantes : Classe String :

Méthode ou fonction	Utilisation
length	attribut retournant la longueur de la chaîne
charAt()	renvoie le caractère qui se trouve à la position spécifiée
indexOf()	première position d'un sous-chaîne dans un chaîne
lastIndexOf()	idem à la précédente mais en commençant par la fin
split()	Transforme une chaîne de caractères en un tableau de chaînes, en découpant la chaîne d'origine selon le séparateur spécifié en argument
substring()	renvoie une sous chaîne spécifié par la position des caractères de début et de fin dans la chaîne initiale
toLowerCase()	conversion en minuscule
toUpperCase()	conversion en majuscule
bold()	affecte l'attribut gras au texte
italics()	affecte l'attribut italique au texte

Nous constatons ici que les chaines de caractères sont des objets complexes incluant le contenu de la chaîne mais aussi des attributs de formatage. Ceci est extrèmement pratique pour un langage complètement dédié à la production de pages WEB.

Un petit exemple de ce qu'il est possible de faire avec cet objet :

```
d = " bonjour "
document.write( " Le troisième caractère de la variable d est " + d.charAt(3) + " <BR> " )
Exemple 2 :
Cet exemple montre encore quelques méthodes complémentaires de formatage des objets de la cl
String.
<HTML>
<HEAD>
<TITLE>L'objet String</TITLE>
<script language="JavaScript">
document.write("Dame Belette, un beau matin" + "<BR>")
document.write("Dame Belette, un beau matin".length)
document.write("<HR>")
annonce="Texte d'essai"
document.write(annonce)
document.write("<BR>" + annonce.big())
document.write("<BR>" + annonce.blink())
document.write("<BR>" + annonce.bold())
document.write("<BR>" + annonce.fixed())
document.write("<BR>" + annonce.italics())
document.write("<BR>" + annonce.bold().italics())
document.write("<BR>" + annonce.small())
document.write("<BR>" + annonce.strike())
document.write("<BR>La valeur de V" + "MAX".sub() + " est 2")
document.write(" La valeur de V" + "MAX".sub().small() + " est 2")
document.write("<BR>La valeur de V" + "3".sup() + " est 343")
document.write(" La valeur de V" + "3".sup().small() + " est 343")
document.write("<HR>")
document.write("Un carré est une circonférence qui a mal tourné" +
```

```
" (Pierre Dac)".fontsize(4) + "<BR>")
for (i=1; i<=7; i++)
{ taille = i
document.write(("fontsize(" + taille + ")").fontsize(i) + "<BR>")
}
</script>
</HEAD>
<BODY>
<HR>
<H3 ALIGN=CENTER>Nous sommes ici dans la section &lt;BODY&gt;</H3>
<KBD>Texte d'essai avec KBD</KBD>
<BR>
<S>Texte d'essai avec S</S>
</BODY>
</HTML>
```

• Des objets en pagaille !

Javascript est un langage résolument objet, même s'il permet un tas de choses que les bons langages objets interdisent, comme par exemple, l'accès directe aux attributs des classes. Mon exposé se limitera ici à la manière dont on peut créer des objets en Javascript :

On créé des instances d'objets à l'aide de l'instruction new et de l'appel à leur constructeur.

```
quidam = new personne("gouinaud","christophe");
```

Donc définir de nouveaux objets, c'est définir de nouveaux constructeurs avec le mot réservé this. L'objet précédent se créé donc comme :

```
function personne(nom,prenom,daten)
{
this.nom = nom;
this.prenom = prenom;
this.date = new date(daten);
}
```

Ca paraît un tantinet bizarre, mais sa facilite l'interprétation du code. L'ajout d'une méthode se fait en deux temps : création de la méthode puis ajout de la méthode à l'objet.

```
function age()
{
cdate = new date();
ns = cdate-this.date;
age =  ns/(3600*24*365);
return(age);
}

function personne(nom,prenom)
{
this.nom = nom;
this.prenom = prenom;
this.date = new date(daten);
this.age = age;   // on met la methode
}
```

□

FIG. 4 – Hiérarchie d'objets du navigateur.

Nous remarquons que la généricité des méthodes ne devrait pas poser trop de problèmes.

• Les objets du navigateur !

L'avantage de javascript par rapport aux autres langages de script est que, pour lui, tous les éléments d'une page html sont des objets. Pour pouvoir les manipuler, il suffit d'utiliser soit les objets préfinis ou de les nommer. La hiérarchie d'objets par défaut est représentée sur la figure 3.5.4

Nous constatons donc que nous pouvons manipuler chaque élément de la page par des références à cette hiérarchie. Cependant, il est souvent plus pratique de nommer les objets en utilisant l'attribut name.

```
<IMG SRC="toto.gif" NAME="toto">
```

Les objets ont ensuite différents attributs et reçoivent différents événements qui leurs sont propres. Nous verrons dans les formulaires qu'il est possible de modifier leur comportement pour les rendre plus dynamiques.

• Des événements comme s'il en pleuvait !

Qui dit page Web dit utilisation d'un écran, d'un clavier et souvent d'une souris sur le poste de visualisation. La programmation en javascript permet de gérer soit-même la réponse aux événements perçus par le navigateur et donc de modifier dynamiquement les éléments de la page Web en fonction des actions de l'utilisateur.

Le nombre des événements reconnu par javascript est limité, mais s'étend de version en version. Leur nom commence toujours par on (à l'occasion de). En voici une liste avec une courte explication et le nom des classes qui les génèrent.

Evénement	Appelé lorsque	objets
onAbort	L'utilisateur stoppe le chargement d'une image	Image
onBlur	Un élément d'entrée perd le focus	select, text, textarea
onChange	Identique à onBlur, mais le contenu a changé	select, text, textarea
onClick	L'utilisateur clique sur un élément du formulaire	bouton, radio, checkbox, reset, submit
onError	Une erreur a eu lieu pendant le chargement du document	form
onLoad	Une fenêtre ou frameset vient d'être chargé	frame ou body
onMouseOut	La souris quitte un lien ou une zone	link, et éléments de formulaire
onMouseOver	La souris se trouve sur un lien ou une zone	idem
onReset	L'utilisateur clique sur le bouton RESET	form
onSelect	Le texte d'une boîte de saisie a été sélectionnée	text, textarea
onSubmit	L'utilisateur clique sur le bouton SUBMIT	form
onUnload	Le contenu d'une fenêtre est effacé	frame ou frameset

Ces événements s'emploient ensuite soit par l'ajout d'un attribut à leur nom dans la balise HTML de l'objet visé. Par exemple pour modifier le chargement d'une page

```
<BODY onLoad="QueLaLumiereSoit();">
```

où la fonction QueLaLumiereSoit est le callback de l'événement et doit être bien évidemment écrite par mimile !

• Des formulaires intelligents !

Javascript ne comporte pas de fonctions permettant le dialogue client serveur, son utilisation dans les formulaires sera donc limitée au contrôle des données et à la modification dynamique des formulaires. Ce langage permet, par exemple, de mettre en évidence les erreurs de saisie de l'utilisateur (numéro de téléphone), le reformatage de données (nom sous la forme Gouinaud), la gestion dynamique des listes et l'autocomplétion de certains champs (code postal, ville).

Pour javascript, chaque formulaire est vu comme une instance d'objet

```
<HTML>
  <HEAD> <TITLE>Essai de formulaire</TITLE> </HEAD>
<BODY bgcolor = "#fef0e0">
<FORM NAME="rens" METHOD="GET" ACTION="http://localhost/cgi-bin/fcgi">
<H4>Identification de la personne :</H4>
<TABLE>
  <TR>
    <TD> Nom:<td>
<TD><INPUT TYPE="text" NAME="nom" onChange="rens.nom.value=rens.nom.value.toUpperCase();"
Size=15></TD>
   </TR>
   <TR><TD>Prenom : </TD>
       <td></TD>
    </tr>
</TABLE>
<SCRIPT LANGUAGE="javascript">
function ditville()
{
ville="";
switch (document.rens.codepostal.value)
 {
   case "63000": ville="Clermont-Ferrand";break;
   case "63170": ville="Aubiere"; break;
   default: break;
  }
document.rens.ville.value=ville;
}
</SCRIPT>
<TD><INPUT TYPE="text" NAME="codepostal" Size="5" onChange="ditville();"></TD>
<TD><INPUT TYPE="text" NAME="ville" Size="30" ></TD>
<INPUT TYPE="submit" VALUE="OK"><INPUT TYPE="reset" VALUE="Effacement">
</FORM>
</HTML>
```

Dans cet exemple, on met en majuscule le nom une fois qu'il a été tapé et on remplit le champ ville une fois le code postal entré.

Le nom de l'objet formulaire est la valeur de l'attribut NAME spécifié dans le code HTML. Les champs du formulaire sont vus comme des objets attribut de l'instance du formulaire. Il est donc facile de les manipuler en modifiant leur valeur (attribut value en général). Comme cela est fait dans l'exemple ci-dessus, on utilise le gestionnaire d'événements pour détecter les modifications et réagir en conséquence.

Le tableau suivant donne une description des objets relatifs aux formulaires :

Classe	Attribut	Description
forms	elements[]	Tableau instantiant les différents éléments du formulaire
	length	Nombre d'éléments dans le tableau forms
	method	Valeur de l'attribut METHOD
	target	Valeur de l'attribut TARGET
button	name	Nom au sens Javascript du bouton. Il s'agit de la valeur de l'attribut NAME
	value	Texte affiché dans le bouton rectangulaire, attribut VALUE
	click()	Simule un click sur le bouton ONCLICK= " Traitement "
checkbox	checked	passe à true lorsque la case est cochée (lecture seule)
	defaultChecked	valeur par défaut de la coche
	name	nom de l'objet
	type	C'est le type de cet objet, checkbox
	value	Sa valeur
	événements	ONCLICK= " Traitement "
radio		Les propriétés sont les mêmes que celles de l'objet checkbox. Il y a en plus length, qui retourne le nombre de Chekbox dans un objet radio.
select	defaultSelected	Valeur de la sélection initiale
	length	Nombre d'éléments dans la liste
	name	son nom
	options	C'est un tableau d'objets Options dont chaque élément représente l'une des options de la liste.
	selectedIndex	C'est le rang (compté à partir de 0) de l'article de la liste qui a été sélectionné par l'utilisateur (lecture seule)
	type	Indique s'il s'agit d'une sélection unique (select-one) ou d'une sélection multiple (select-multiple), blur() Enlève le focus à l'objet select concerné focus() Inverse de blur()
	événements	ONBLUR= " Traitement " ONCHANGE= " Traitement " ONFOCUS= " Traitement "
text	defaultValue	Valeur de l'attribut VALUE
	name	son nom
	type	text
	value	Correspond au contenu tapé par l'utilisateur
	Méthode	blur(), focus(), select(), Simule la sélection du texte. Malheureusement, il n'y a aucune possibilité de retourner le texte sélectionné.
	événement	ONBLUR= " Traitement " ONCHANGE= " Traitement " ONFOCUS = " Traitement " ONSELECT= " Traitement "

Le principal problème que pose l'utilisation de javascript pour l'autocomplétion des champs est le problème et l'importation des bases de données nécessaires. Si, par exemple, on importe l'ensemble des codes postaux de France en une opération, le fichier généré sera de l'ordre de 500 Ko et par conséquent la durée du chargement de la page prohibitive. L'astuce la plus simple est d'utiliser plusieurs frames réalisant une partie des opérations via du PHP.

• Conclusion

Javascript est un langage d'enrichissement de pages permettant de gérer efficacement du côté du browseur des actions complexes. Sa structuration objet permet une programmation claire et une bonne gestion des événements. Son principal écueil est la compatibilité des navigateurs. Il faut aussi noter que beaucoup de sites commerciaux abusent de cet outil pour forcer l'utilisateur à recevoir des pubs et que, par conséquent, beaucoup de personnes interdisent à leur navigateur d'interpréter ce genre de script.