HORS-SÉRIE 9

Décembre/ Janvier 2001

LINUX FRANCE MAGAZINE

& HURD

France Métro 39 F/5,95 € - BEL 260 FB/6,45 € 39 F - 5,95 €

Installer son serveur web à la maison

Dépôt du nom de domaine et enregistrement DNS

Installer votre serveur et configurer le réseau

Protéger votre machine

Ouvrir le serveur HTTP

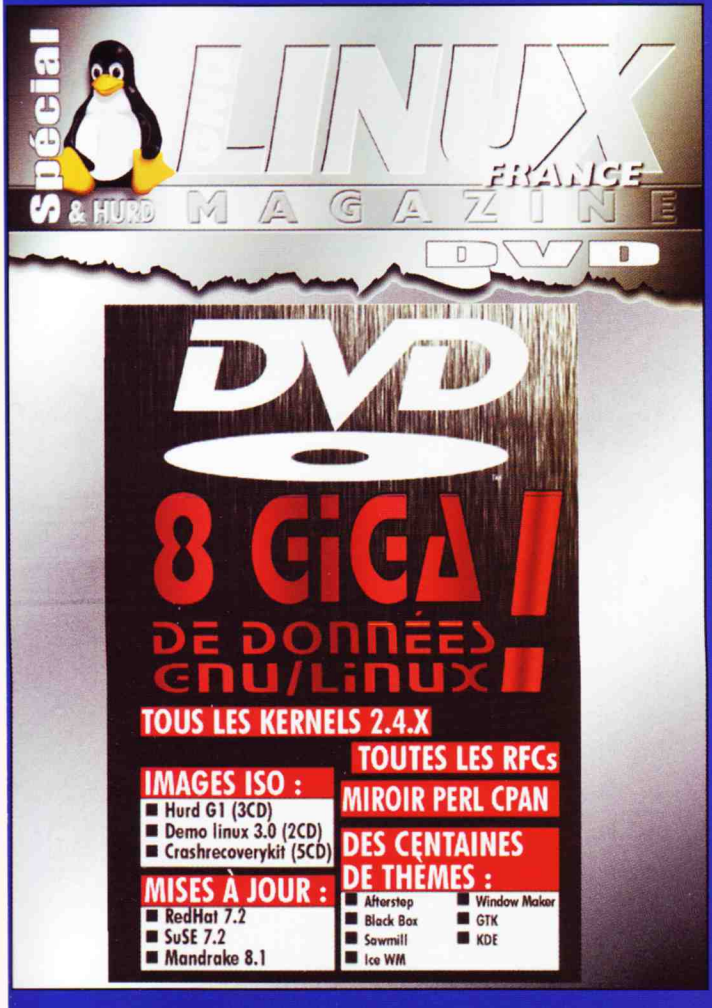
Utiliser PHP et MySQL pour vos pages

L 5066 - 9 - 39,00 F - 5,95 € - RD

En vente le 11 janvier chez votre marchand de journaux

La sécurité pour tous les systèmes enfin traités de manière technique et professionnelle

En vente fin novembre chez votre marchand de journaux



Linux Magazine DVD

TOUS LES KERNELS 2.4 X	MISES À JOUR : RedHat 7.2, SuSE 7.2, Mandrake 8.1
IMAGES ISO : Hurd G1 (3CD), Demo linux 3.0 (2CD), Crashrecoverykit (5CD)	TOUTES LES RFCs :
	MIROIR PERL CPAN
	DES CENTAINES DE THÈMES

Edito

Bonjour,
et bienvenue dans ce numéro hors série de Linux Mag consacré à l'installation d'un serveur Web à la maison (ou au bureau).

Les connexions permanentes à Internet sont de plus en plus populaires. Elles constituent un confort de connexion non négligeable, mais le connaisseur y verra un autre avantage : la possibilité d'ouvrir sa propre auberge dans le village virtuel planétaire qu'est Internet. Jusqu'à présent, les utilisateurs souhaitant diffuser de la connaissance et créer leur propre site Web étaient limités aux offres d'hébergement gratuit ou payant de certains fournisseurs.

Les possibilités offertes par ce type de service sont nombreuses, mais un point essentiel était exclus : le contrôle total du système. La dépendance vis-à-vis de l'hébergeur limite en effet la liberté du responsable du site. L'absence de liberté quant au logiciel utilisé est cependant contre-balancée par le fait que toute la maintenance logicielle est laissée à la discrétion de l'hébergeur. Il n'en reste pas moins que la présence physique d'une machine étant un véritable serveur Internet est quelque chose de vraiment grisant :)

Le magazine que vous tenez dans les mains a pour but de vous fournir toutes les indications de base nécessaires pour bien débuter avec votre serveur Web. Nous y verrons étape par étape et de manière pratique comment installer et configurer votre machine. Vous apprendrez à configurer la sécurité de votre système et un serveur HTTP, puis comment utiliser les ressources les plus souples pour créer votre ou vos site(s) avec PHP et MySQL.

Enfin, certains articles vous mettront le pied à l'étrier en ce qui concerne des points précis dont la mise en place d'une connexion ADSL, câble, ou encore comment utiliser SSL et votre serveur Apache.

Les applications choisies pour les articles sont les classiques du genre, à commencer par Apache. Bien sûr, il existe d'autres logiciels, d'autres méthodes et d'autres concepts que serez à même de juger après avoir créé votre tout premier site...

Précisons qu'une connaissance du système GNU/Linux est requise en ce qui concerne le fonctionnement en réseau et les principaux utilitaires (navigation dans les répertoires et édition des fichiers).

En guise de mot de la fin, je tiens à remercier chaleureusement Vincent Renardias pour sa participation sans faille dans ce hors série (filtrage).

Bonne lecture à tous et bienvenue dans votre nouveau cyber chez-vous...

Denis Bodor

Dépôt de nom de domaine

Installation de base

Web/HTTP

Anciens numéros et bon de commande

Déposer son nom de domaine

Avant de vous lancer dans l'installation et la configuration d'une machine qui deviendra votre serveur, vous devez lui choisir un nom et un domaine. Il s'agit de son nom public tel que les visiteurs de votre ou vos sites vont le voir et l'utiliser. Ce nom ne peut être choisi arbitrairement. Hors de question en effet de décréter du jour au lendemain que votre machine est www.linux.org !

Qu'est-ce qu'un nom de domaine ?

Lorsqu'une machine est connectée à Internet, elle se voit attribuer un numéro, ou plus exactement une adresse IP. Si vous possédez des services fonctionnant sur cette machine (comme par exemple un httpd), les gens peuvent, de l'extérieur, accéder à votre serveur en utilisant cette adresse IP. Dans le cas d'un serveur HTTP, le client utilisera la syntaxe *http://www.bbb.ccc.ddd/* dans son navigateur. Cela n'est, bien sûr, pas très élégant : une adresse IP n'est pas facile à retenir pour un utilisateur. De plus, une même machine peut parfaitement héberger plusieurs serveurs sous des noms différents (c'est ce qu'on appelle les hôtes virtuels). Pour corriger le problème, on associe un nom unique dont vous êtes le seul détenteur à votre machine connectée à Internet : le nom de domaine.

Le nom de domaine est composé de deux éléments :

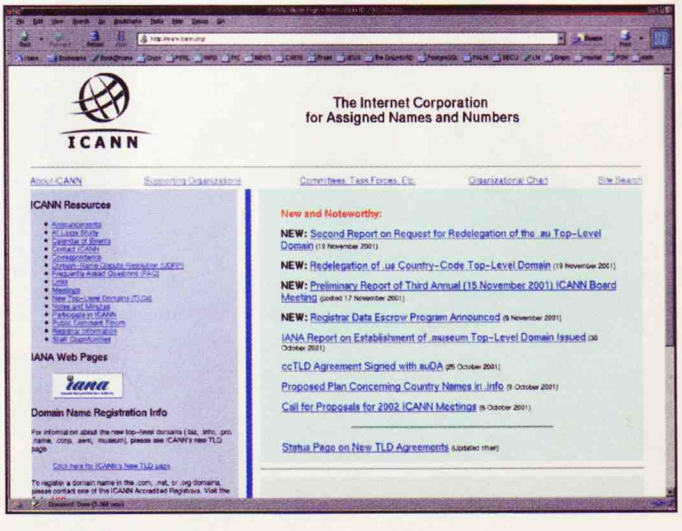
- Le TLD (*Top Level Domain*) est le suffixe du nom de domaine (.org, .net, .fr, .com, etc.). On peut classer les TLD en deux catégories ; dans un premier temps, ceux qui définissent un lieu géographique (.fr, .de, .jp). Dans le cas du .fr, les manipulations ne sont pas les mêmes et vous ne pourrez déposer un .fr que sous certaines conditions. Ensuite, viennent les TLD internationaux (.com, .net, .org, .firm, .shop, .web, .arts, .rec, .info, et .name). Ceux-ci peuvent être librement utilisables et habituellement beaucoup moins chers. Notez que seuls les .com, .net, et .org sont actuellement généralement utilisés. L'organisme contrôlant la validité des TLD, l'ICANN, poursuit, à l'heure où nous écrivons ces lignes, les procédures pour les autres TLD. La mise en œuvre effective des nouveaux TLD suit une procédure par phase. Les .biz et les .info ont passé les trois phases et les organismes de dépôt de noms de domaines commencent à être accrédités. Pour ce qui est du .name (name est à remplacer par votre nom), la phase 3 est prévue pour le premier trimestre 2002.

- Un nom arbitrairement défini par le dépositaire du nom de domaine dans la mesure où le nom de domaine complet n'est pas encore déposé par quelqu'un d'autre. Vous ne pouvez pas choisir n'importe quoi dans cette partie. Les caractères acceptés sont :

  - les caractères de l'alphabet latin (a à z). La casse n'a pas d'importance, les majuscules et les minuscules sont traitées de manière équivalente.

  - les chiffres dit arabes (0 à 9) ;

  - le symbole "-" (trait d'union ou symbole de soustraction) peut être utilisé mais est interdit en début et en fin de nom.



Tous les autres symboles sont interdits pour le moment. Mais sachez que l'ICANN travaille sur une internationalisation des noms de domaines avec utilisation des jeux de caractères Unicode (UTF-8).

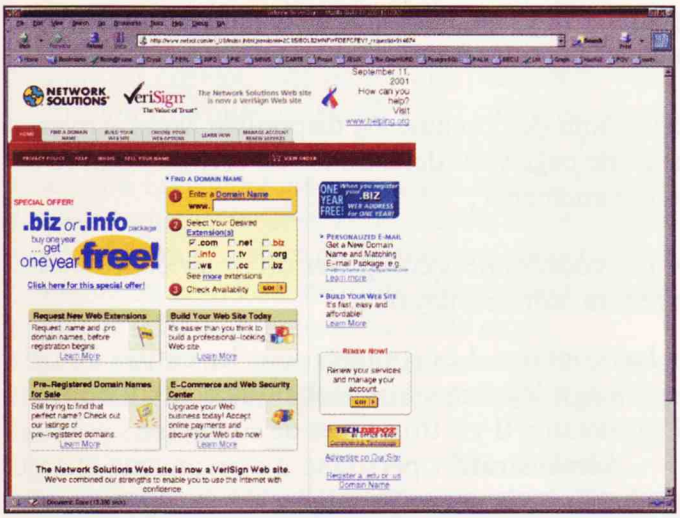
Il est donc probable que d'ici quelques années, vous voyiez apparaître des noms de domaines accentués ou encore en caractères Kanji ou cyrillique. Ceci est cependant une réforme à grande échelle et un véritable travail de titan en termes d'organisation.

Déposer un nom de domaine

La première étape consiste à choisir un nom significatif pour votre domaine. Voici quelques recommandations :

- Plus un nom de domaine est court, plus il est facile à retenir.

- Choisissez un nom prononçable composé de syllabes. Un nom comme xc56-3a7d.net n'est certainement pas encore déposé, mais imaginez-vous le dictant au téléphone à un ami ou un collaborateur...

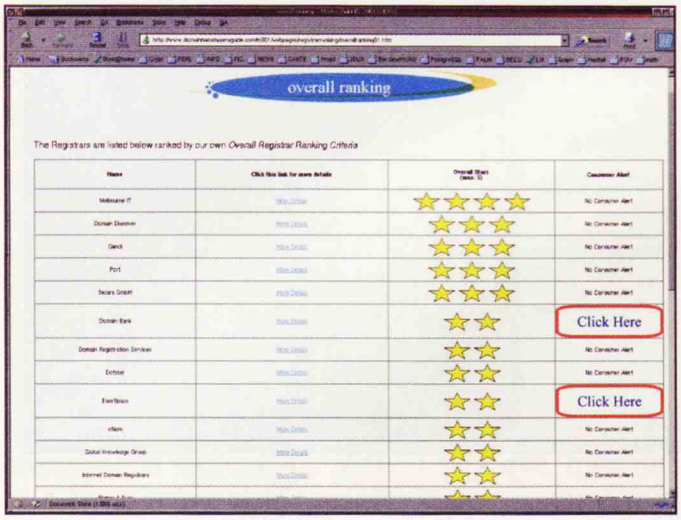


- Faites au plus proche du sujet du site. Si vous comptez mettre en place un serveur HTTP (ou autre) pour un projet personnel (logiciel ou autre), choisissez tout simplement le nom de votre projet comme nom de domaine. A l'inverse, si vous désirez regrouper tous vos travaux et vos données personnelles sur le nouveau site, choisissez un nom de domaine qui est fortement lié à votre personne (nom de famille, pseudo, *nickname*). Vous aurez tout loisir ensuite de décliner ce nom de domaine en plusieurs machines (www.mondomaine.org, projet1.mondomaine.org, docs.mondomaine.org, etc.).

- Evitez absolument les noms de domaines faisant référence à une marque déposée ou simplement incluant de telles choses dans le nom complet. Des cas se sont déjà présentés par le passé où le propriétaire du nom de domaine a été obligé de le céder au détenteur de la marque. Nous ne nous étalerons pas ici sur le bien-fondé de telles démarches... Evitez simplement les ennuis !

Comme nous l'avons dit plus haut, il faut, bien sûr, que le nom de domaine ne soit pas encore déposé par une autre personne. D'autre part, c'est rarement une bonne idée de déposer mondomaine.com si mondomaine.org est déjà pris. Non seulement vous risquez d'avoir des visites sur votre site de la part de personnes intéressées à l'autre projet, mais en plus, vous risquez de vous attirer les foudres du propriétaire de l'autre domaine (et c'est souvent compréhensible).

Un dépôt de nom de domaine n'a rien à voir avec l'association à votre adresse IP. Ceci est une étape différente. Le dépôt du nom de domaine est une réservation de ce nom à titre personnel. En clair, vous pouvez parfaitement déposer un nom sans pour autant avoir à ce moment de machine prête à accueillir un site. Il est important de bien comprendre ce point : le dépôt d'un nom de domaine est la location d'un droit exclusif de l'utiliser. Il est intéressant pour une société de déposer son nom de domaine même si elle ne possède pas encore l'infrastructure informatique nécessaire à la mise en place d'un serveur. Elle réserve ainsi son territoire sur le Net.

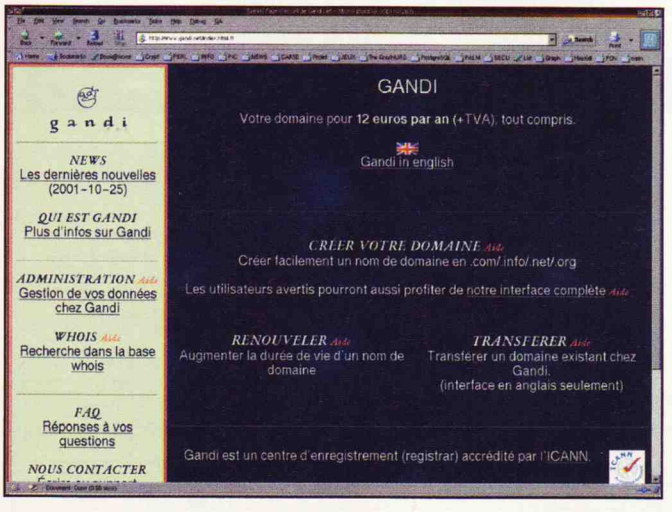


Là encore, certaines personnes ou entreprises abusent. Elles déposent tous les noms de domaines qu'elles jugent intéressants à plus ou moins long terme et attendent simplement qu'une personne souhaite l'utiliser. A ce moment-là, elles négocient la cession du nom à un tarif plus élevé qu'un dépôt. On ne peut, bien évidemment, que dénoncer ce genre de pratique, mais il faut faire avec...

Dépôt de nom de domaine

Le dépôt d'un nom de domaine se fait par l'intermédiaire d'un *registrar* (bureau d'enregistrement en français) agréé ICANN (pour les .com, .net et .org). Le travail de l'ICANN consiste en effet à déterminer des règles de nommage, des TLD valides et un grand nombre d'autres recommandations. Mais l'ICANN ne procède pas directement à l'enregistrement de noms de domaines. Cela est laissé à la discrétion d'entreprises (les registrars) qui servent d'intermédiaires entre vous et l'ICANN. Il existe un grand nombre de registrars en France et à l'étranger. Jusqu'à il y a un peu moins de deux ans, les registrars se livraient à une guerre sans merci sur les prix et les services. On pouvait jusqu'alors facilement se faire avoir par des packs tout compris (routage, DNS, dépôt de nom, hébergement) alors que seul le dépôt du nom nous intéressait.

La concurrence est toujours de mise mais un nouveau registrar a fait son apparition en mars 2000 : GANDI. Loin de nous l'idée de faire la publicité d'une entreprise, mais GANDI a quelque chose de plus par rapport aux autres. GANDI est une société française créée par Pierre Beyssac, Laurent Chemla, David Nahmias et Valentin Lacambre. Ce dernier est également responsable du très connu altern.org.



Mais que possède donc GANDI que les autres n'ont pas ? Ceci : GANDI est orienté principalement vers les particuliers et les associations avec la volonté de fournir à tout individu facilement et au prix le plus bas possible des noms de domaine. Vous l'aurez compris, le but de cette société n'est pas le profit maximum mais d'appliquer une politique d'ouverture permettant à tous de posséder son nom de domaine.

Procédure avec GANDI

La procédure pour enregistrer/déposer son nom de domaine est relativement simple. Il vous suffit d'un accès Internet (qui en aurait douté), d'un navigateur supportant SSL et d'un moyen de paiement par carte bancaire. Allez tout simplement sur **http://www.gandi.net/index.html.fr** et choisissez l'option "Votre domaine". Vous arriverez normalement sur une page vous demandant de saisir le nom de domaine que vous souhaitez enregistrer et le TLD à utiliser (pour l'heure .com, .org ou .net).

Si le nom de domaine choisi est déjà enregistré, vous obtiendrez un message signalant le problème et un lien vous informera sur son propriétaire. Ce lien vous permettra d'obtenir toutes les informations utiles sur le domaine et la personne au cas où vous souhaiteriez lui racheter son domaine. Dans la plupart des cas, mieux vaut revenir en arrière est choisir un autre nom.

Si le nom de domaine est disponible, vous arriverez sur une page vous demandant un certain nombre de renseignements :

- Les coordonnées complètes du propriétaire (nom, prénom, adresse, pays).

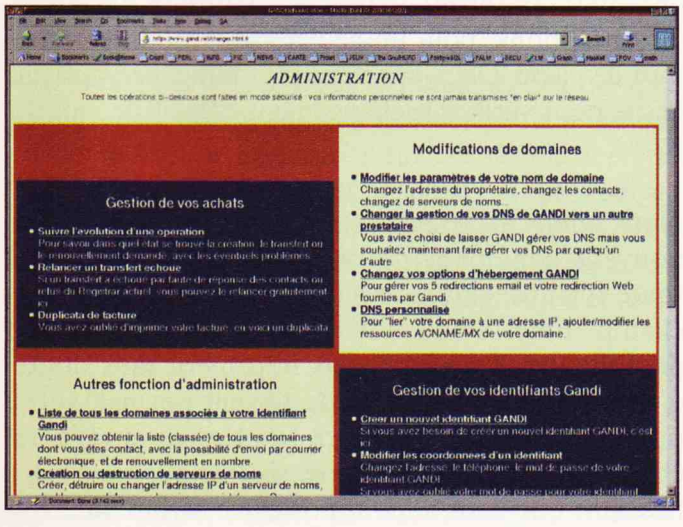
- Les contacts. Les contacts sont les entités autorisées à agir légalement ou techniquement au nom du Propriétaire. Il y a trois types de contacts :
  - **Administratif** : personne ou organisme chargé de répondre aux questions légales au sujet du domaine. C'est en général vous-même ou une personne autorisée de votre société.
  - **Technique** : personne ou organisme chargé de prendre les décisions techniques au sujet du domaine.
  - **Facturation** : personne ou organisme qui reçoit les factures d'enregistrement et de renouvellement.

  Seul le contact administratif est obligatoire. Vous pouvez vous passer de spécifier un contact de facturation ou technique et dans ce cas, le contact administratif sera utilisé en lieu et place. Si vous restez dans le cadre de ce hors série, vous ne spécifierez qu'un contact administratif puisque possédant la machine serveur chez vous, vous êtes responsable des trois éléments.

Les contacts sont indexés par un identifiant. Il s'agit d'une référence interne à GANDI. En princi-

pe, vous ne devez pas encore posséder d'identifiant GANDI (puisque sinon vous ne liriez pas cet article). Pour créer votre identifiant, cliquez sur le mot "ici" à droite du champ "Identifiant du contact administratif:". Vous arriverez sur un formulaire vous demandant de saisir différentes informations. Si vous êtes tenté de mettre n'importe quoi dans ce formulaire, abstenez-vous ! Le contact est non seulement la personne responsable (en fonction du type de contact), mais c'est également elle qui sera contactée en cas de problème (technique ou de gestion). Pour sécuriser l'utilisation de cet identifiant, un mot de passe vous est demandé. Là encore, saisissez quelque chose de cohérent. Un bon mot de passe est un mélange de chiffres et de lettres sans signification particulière, mais dont vous pouvez facilement vous souvenir. La plupart des champs sont obligatoires ; vous recevrez un message d'avertissement si un ou plusieurs de ces champs ne sont pas renseignés.

Si tout est correct, une page vous présentera votre identifiant nouvellement créé. Il est composé de vos initiales et d'un chiffre suivi de "-GANDI". Celui-ci vous aura également été envoyé par email à l'adresse que vous aurez spécifiée. Pour remplir automatiquement le champ contact du formulaire concernant le domaine en cours de création, cliquez tout simplement sur le bouton présent sur cette page.
Laissons cela en suspens pour le moment pour nous intéresser aux bases nécessaires à l'utilisation des derniers champs...



DNS

L'enregistrement d'un nom de domaine est uniquement une association entre votre personne et ce nom. Pour faire la liaison entre le nom de domaine et votre machine connectée à Internet, il vous faut utiliser un *Domain Name Server* (DNS). Nous n'allons pas ici faire un cours sur le fonctionnement des DNS et de la résolution des noms de domaines. Sachez simplement qu'un seul serveur fait autorité pour donner la correspondance entre IP et nom de domaine. Cependant, tous les serveurs de la planète partagent une base de connaissance via un mécanisme complexe. Il vous suffit donc de renseigner un seul DNS pour que (après une période de 24 à 48 heures) l'ensemble des visiteurs dè votre site connaissent la correspondance IP/nom.

La plupart des registrars proposent de vous enregistrer dans leur DNS. Cela fait partie des services accompagnant la prestation d'enregistrement de nom de domaine. Deux solutions s'offrent à vous, chacune ayant ses avantages et ses inconvénients :

- La redirection : Vous possédez déjà un serveur Web hébergé et une ou plusieurs adresses email. Dans ce cas, vous pouvez définir que toutes les connexions arrivant sur le nouveau domaine seront redirigées vers, par exemple, votre page perso chez votre FAI. Il en va de même pour les emails où vous définirez que les mails arrivant sur moi@mondomaine.org seront renvoyés automatiquement vers l'adresse que vous aurez spécifiée pour l'enregistrement du nom de domaine. Cette méthode présente l'avantage d'être facile à mettre en œuvre, mais peut poser des problèmes dans le cas où vous hébergez sur votre machine personnelle plusieurs sites. En effet, imaginez que vous ayez enregistré les noms de domaines *nomdomaine1.org* et *mondomaine2.org* et que vous redirigiez ces deux domaines vers votre machine. Le serveur Web est capable de travailler avec des domaines virtuels (plusieurs domaines, et donc plusieurs sites, sur une seule et même machine), mais avec la redirection, il sera incapable de déterminer quel domaine a été demandé originellement par le visiteur. Le résultat sera que, quel que soit le domaine demandé, les visiteurs arriveront toujours sur les mêmes pages...

- L'enregistrement d'une entrée dans un DNS. Là, il ne s'agit plus de redirection. Lorsqu'un visiteur demandera à consulter votre site par son nom de domaine, le DNS de son fournisseur d'accès l'informera de l'adresse IP correspondante (résolution de nom de domaine). L'ajout d'entrées dans un DNS

nécessite quelques connaissances de base mais vous permettra de faire quasiment tout ce que vous voulez.

En ce qui concerne GANDI, c'est à la section 3 du formulaire d'enregistrement de nom de domaine que vous pourrez choisir le service qui vous convient. En cliquant sur "Laisser GANDI s'occuper de ça pour vous en cliquant ICI", vous verrez apparaître une nouvelle page vous permettant de configurer les redirections pour le Web et pour le mail. Les indications fournies sur la page sont relativement faciles à comprendre.

Si, en revanche, vous désirez utiliser une véritable résolution DNS, GANDI vous demande le nom et l'adresse IP du serveur DNS, mais ne propose pas d'hébergement DNS. Vous devrez faire appel à un autre prestataire. En attendant, laissez cette partie vide pour le moment.

ZoneEdit

Là encore, il existe un grand nombre de fournisseurs qui se proposent d'être votre DNS de référence. La quasi-totalité de ces fournisseurs font payer ce service, mais il en est qui proposent (sous certaines conditions) un service de DNS public. Entendez par là, gratuit. Le plus connu est sans doute ZoneEdit.com. Ce prestataire vous permet d'avoir gratuitement un DNS pour un maximum de 5 domaines. Au-delà de cette limite, le service devient payant.

Le principe de la résolution de noms de domaines permet de fixer une adresse IP par machine dans votre domaine. Nous allons voir tout cela par la pratique à l'aide d'un exemple concret basé sur les services gratuits mis à disposition par ZoneEdit.

La première étape consiste à ouvrir un compte chez le fournisseur de service. Allez tout simplement sur la page d'accueil (**www.zoneedit.org**) et choisissez le lien "Free Sign Up" (en haut à gauche). Vous arriverez sur une page vous demandant d'entrer vos coordonnées. ***NE METTEZ PAS N'IMPORTE QUOI*** ! Des informations non valides ou s'avérant inexactes conduiront à la suppression pure et simple du compte. Soyez correct et respectez ces indications, c'est la moindre des choses. De plus, si par la suite, vous êtes satisfait du service et que votre site devient populaire, payez le service. Note : ZoneEdit signale que vos coordonnées ne seront pas utilisées pour de la diffusion publicitaire, ce qui n'est pas à négliger.

Après avoir renseigné tous les champs et validé le formulaire, vous recevrez, par email, un identifiant (login) et un mot de passe. Par mesure de sécurité, la première chose à faire est de changer votre mot de passe. En effet, celui généré par ZoneEdit vous a été transmis par email en clair. De plus, il est généralement assez difficile à retenir. Revenez donc sur la page d'accueil du site et loguez-vous via le lien "Login" (en bas à gauche). Entrez identifiant et mot de passe. Vous arriverez alors sur une page minimaliste :

Dans le menu supérieur

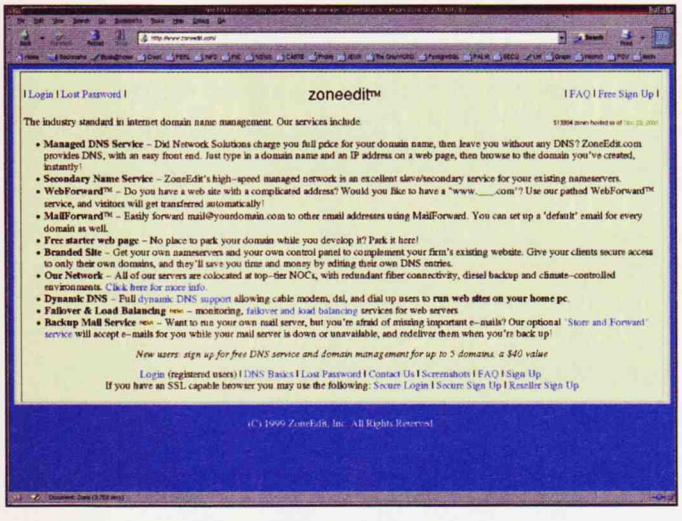
- *Edit Zone* : vous permet d'éditer les informations relatives aux zones (domaines) ;
- *User Options* : vous permet d'éditer les informations vous concernant ;
- *Account Info* : liste les informations relatives à la gestion de votre compte (zones gratuites, limite de gratuité) ;
- *Add Zones* : ajoute une nouvelle zone (domaine) ;
- *Help* : aide en ligne.

Dans la partie centrale, à ce niveau, vous ne devez avoir qu'un lien "Add Zones". Par la suite, les zones enregistrées seront listées ici. Cliquez donc sur "User Options" pour immédiatement modifier votre mot de passe. Déloguez-vous puis reloguez-vous sur le site pour vérifier le changement effectif de mot de passe.

Nous allons à présent nous attacher à ajouter une nouvelle zone pour notre domaine. Pour cela, en toute logique, suivez le lien "Add Zones". Dans le champ "Enter Domain Name:", entrez votre... nom de domaine. ZoneEdit vous informera sans doute d'un problème. Ceci est parfaitement normal, votre domaine n'étant pas encore totalement enregistré chez GANDI, ZoneEdit vous signale que les modifications que vous allez apporter ne seront valables qu'environ 72 heures après l'enregistrement auprès du registrar. Dans la page d'avertissement, il est fait mention de deux lignes concernant le serveur de nom (DNS) :

Primary Nameserver: ns1.zoneedit.com (207.228.252.101)
Secondary Nameserver: ns3.zoneedit.com (209.61.140.1)

Ces lignes peuvent être différentes pour vous. Il s'agit très exactement des informations à ajouter dans la section 3 du formulaire d'enregistrement de GANDI : faites donc un simple copier/coller de ces lignes dans les champs adéquats et payez votre dépôt de nom de domaine. La phase Gandi est terminée.



Cliquez ensuite sur le lien en bas à droite ("Start editing your domain before it goes live") pour poursuivre les manipulations. Un résumé des informations concernant la zone en cours d'édition va apparaître. Les informations sont les suivantes :

- *IP Addresses* : Vide pour le moment, c'est ici que seront renseignées les adresses IP de chaque machine du domaine. En principe, pour un simple serveur Web, une seule adresse est spécifiée. Nous n'entrerons pas dans le détail, mais habituellement, une autre machine est précisée en tant que serveur de mail. Nous verrons comment renseigner ce champ plus loin.

- *Mail server* : Permet de renseigner sur le nom de la machine qui gèrera les mails pour votre domaine. Le présent magazine ne couvre pas cet aspect des serveurs, qui sera traité dans un prochain numéro. Nous reviendrons sur la question à ce moment.

- *WebForward* : Permet d'opérer le même travail que le service de redirection de GANDI.

- *MailForward* : Idem pour le mail.

- *WebPark* : Il s'agit d'un système spécifique à ZoneEdit permettant, en l'absence d'infrastructure réseau, de permettre l'utilisation du domaine et de sa résolution de nom. En fait, il s'agit simplement d'afficher une page d'information au visiteur en attendant la mise en place d'un véritable site. Les bannières sont en anglais et le choix limité à "A vendre" et "En construction".

- *Nameserver* : Permet de changer le DNS ZoneEdit à utiliser. Il n'est pas nécessaire de faire la moindre modification ici.

Voyons à présent point par point ce dont nous avons besoin...

1) Adresse IP
Ceci n'est utilisable que dans le cas où l'adresse IP de votre machine destinée à devenir un serveur Web est statique. C'est-à-dire que si vous déconnectez la machine d'Internet et coupez la connexion avec le fournisseur d'accès et que vous vous reconnectiez, votre adresse IP ne changera pas. C'est le cas, par exemple, de l'offre ADSL de Nerim. Les connexions câble et celles d'autres fournisseurs d'accès ADSL utilisent plutôt l'adressage IP dynamique (votre adresse change en cas de déconnexion/reconnexion). Renseignez-vous auprès de votre fournisseur pour connaître le type d'adressage fourni et les modalités de changement.

Si votre adresse IP est fixe, cliquez sur "IP Addresses". Un formulaire vous demande un nom de machine et l'adresse correspondante. Le mécanisme ZoneEdit est très bien conçu et vous n'avez, pour un simple serveur Web, qu'à renseigner le champ "adresse". Dès validation, un message de confirmation vous demandera si vous souhaitez créer directement des entrées pour *mondomaine.org* et *www.mondomaine.org*. Répondez donc par l'affirmative à cette question.

Vous pourrez ajouter autant de noms de machines que vous le souhaitez. Nous pourrions très bien avoir *www.mondomaine.org* pointant vers une IP, *toto.mondomaine.org* vers une autre, ou encore *titi.mondomaine.org* vers une troisième adresse. Il est également possible de faire pointer plusieurs noms de machines dans le même domaine (ou dans des

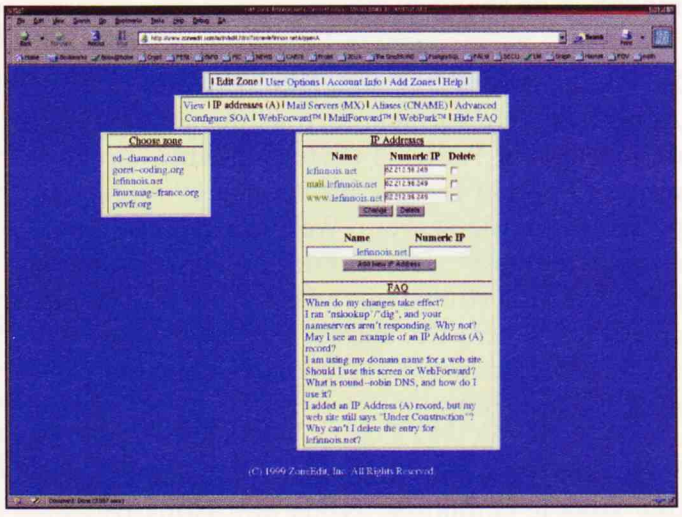
domaines différents) sur la même adresse IP. A la charge du serveur HTTP de faire ensuite le distinguo via des hôtes virtuels (voir article sur Apache).

2) CNAME

Un CNAME est un alias. Vous ne précisez pas d'adresse IP mais un nom de machine complet (comme *www.ailleurs.com*). Cette méthode est à choisir si votre fournisseur ne vous donne pas d'adresse IP fixe. Nous traiterons en détail de cela dans la section DHCP/câble puisqu'il s'agit d'un cas courant avec ce type de connexion. Si vous souhaitez utiliser un CNAME, cliquez sur "Aliases (CNAME)" dans la partie supérieure de la page ZoneEdit. Vous n'aurez ensuite qu'à préciser le nom de la machine (*www* seul, par exemple) et un nom complet pour mettre l'alias en place.

3) Redirection de mail

Dans le présent numéro, nous ne traitons pas de la mise en œuvre d'un serveur de mails. Nous choisirons donc d'utiliser une redirection. En principe, la valeur par défaut lors de la création de la zone doit être correcte. N'importe quel compte (virtuel) utilisé pour envoyer un mail sur le domaine en cours d'édition sera redirigé vers l'adresse email où vous avez reçu votre mot de passe ZoneEdit. Si vous souhaitez faire des changements à ce niveau, il vous suffit de suivre le lien "MailForward" et de supprimer ou ajouter des comptes virtuels (le symbole * permet de spécifier n'importe quel compte virtuel).



Voilà, nous en avons fini avec l'enregistrement du nom de domaine et les zones sur ZoneEdit. Terminons en donnant quelques conseils utiles :

- Vérifiez tout et plusieurs fois ! Même si cela semble pénible, des vérifications répétées vous permettront de trouver et de corriger les erreurs.

- Lisez bien tous les messages donnés avant et après confirmation des modifications (en particulier sur ZoneEdit). Ces messages résument habituellement l'opération en cours ou qui vient d'être effectuée. Une lecture attentive vous permettra d'être sûr des paramètres que vous avez choisis.

- Allez-y doucement ! Même si vous êtes impatient d'ouvrir votre site ou que vous êtes véritablement pris par le temps, vous précipiter conduira à la génération d'erreur. Il n'y a aucune raison valable de gâcher le travail. De plus, un certain temps de réaction est nécessaire avant la mise en œuvre globale. L'enregistrement d'un nom de domaine prend environ 48 à 72 heures. Ajoutez à cela le temps de propagation des modifications de zones (24 à 48 heures), et vous comprendrez que prendre quelques heures en plus pour vous assurer de la validité des informations est préférable à la perte d'une centaine d'heures quelques jours après.

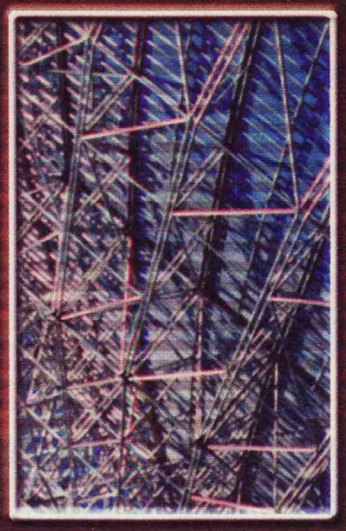
- Soyez responsable. Les fournisseurs dont nous avons parlé dans cet article offrent gratuitement ou à des tarifs préférentiels des services de qualité. Respectez-les ! D'autres prestataires de DNS public (en particulier) ont cessé ce type de services car certains utilisateurs abusaient. Mais respectez également votre propre travail. Renseigner les champs nominatifs avec de fausses coordonnées n'est rien d'autre qu'un acte irresponsable. Soyez adulte, assumez votre travail et vos actes.

- Dernier point : Si vous êtes satisfait, faites-le savoir. Je parle par exemple de ZoneEdit où vous pouvez, sans doute, vous permettre de dépenser quelques dizaines de dollars pour payer un service qui vous satisfait. Pour les autres services qui sont déjà payants, un don, un message de remerciement ou encore une promotion sur votre propre site ne sera pas un mal...

Liens :

GANDI • http://www.gandi.net

ZoneEdit • http://www.zoneedit.com



Bientôt....
Hors série 10
L'aventure continue...

Le Web n'était qu'une étape !
Dans le prochain hors série de Linux Magazine,
vous ajouterez d'autres services à votre serveur :

- le mail avec Exim (filtrage, anti-spam, mailing-lists)
- POP3/IMAP
- du FTP avec proftpd
- un serveur DNS secondaire

et bien d'autres choses indispensables
à un serveur Internet...
Prochainement en kiosque

Installation d'une distribution Debian

Toutes les distributions se veulent les meilleures dans le domaine de la création d'un serveur Internet. Il en est une cependant qui sort du lot de par ses possibilités de gestion de package : Debian. En effet, cette distribution offre un avantage certain dans la manipulation des packages et la gestion des dépendances. Voilà pourquoi le présent magazine de base entièrement sur cette distribution.

Cet article n'est pas une procédure d'installation d'une distribution. Nous allons simplement vous fournir les indications nécessaires constituant un point de départ.

Debian utilise un système de gestion des packages parfaitement adapté au serveur Internet. Une mise à jour d'un serveur Apache, par exemple, sera parfaitement fonctionnelle et les procédures d'arrêt et de redémarrage du serveur seront parfaitement traitées. De plus, si la sécurité est un point important de votre politique de serveur, Debian vous fournira tous les éléments nécessaires en n'utilisant que des logiciels stables et en mettant régulièrement des mises à jour de sécurité sur les serveurs du projet Debian.

Bien sûr, vous n'êtes pas tenu d'utiliser une distribution Debian pour que les indications qui vont suivre dans le présent magazine soient valides. Nous avons fait le choix d'une base de travail, mais toutes les informations dans ces pages sont presque utilisables telles quelles pour n'importe quelle autre distribution GNU/Linux. Il est toutefois recommandé, si vous choisissez une autre distribution, de posséder les connaissances et les habitudes permettant d'adapter les indications. En effet, l'emplacement des fichiers de configuration et les fonctionnalités des différents logiciels changent en fonction du système (modularité de certaines applications).

Préparatifs

Nous avons choisi de ne pas livrer de CD-ROM de distribution avec ce hors série pour plusieurs raisons :

- Si vous ne souhaitez pas installer de distribution Debian mais une SuSE, une Mandrake ou une RedHat, le surcoût en raison de la présence des CD Debian ne serait qu'un point négatif.
- Avec la distribution Debian, vous aurez le choix entre la version stable (*potato*), en test (*woody*) ou beta (*sid*). Il n'y a aucune raison de vous forcer la main quant au niveau de stabilité.
- Vous pouvez parfaitement ne pas avoir besoin d'une quelconque distribution si vous possédez déjà un système GNU/Linux fonctionnel sur votre ou vos ordinateur(s).
- Enfin, partant du principe que si vous souhaitez installer un serveur HTTP, il en découle que vous possédez une connectivité à Internet permanente (câble ou ADSL). Vous pouvez ainsi installer votre distribution via Internet ou encore récupérer les images ISO des CD de manière relativement simple et rapide.

La procédure qui va suivre se base sur une installation via Internet. Si vous ne disposez par encore d'un système GNU/Linux, vous risquez d'être rapidement confronté à un problème. En effet, la connectivité Internet doit être directement utilisable. Ceci n'est pas un réel problème pour Internet par le câble car le système d'installation Debian peut utiliser ce type de connexion directement en lançant un client DHCP. Il en va tout autrement avec l'ADSL et en particulier avec les modems USB.

La procédure d'installation Debian ne permet pas de configurer ce type de connexion lors de l'installation. Vous devez donc disposer d'une machine servant de passerelle Internet. Si votre machine de travail est sous Windows, vous devrez la configurer de manière à ce qu'elle partage la connexion Internet afin de pouvoir installer le serveur. Il vous faudra sans doute installer une application spécifique pour cela.

Si votre machine de travail est déjà sous GNU/Linux, reportez-vous aux articles sur l'ADSL et le filtrage pour configurer la connexion et le partage d'accès.

Enfin, la dernière solution qui se présente est une installation par disquettes. Cela reste pénible et peu sûr (en fonction de la qualité physique du support), mais ce type d'installation s'avèrera être une ultime solution.

Installation

Prévoyez un jeu complet de disquettes pour l'installation. Si vous disposez d'une connectivité directe à Internet, il vous faudra 6 disquettes. Pour une installation entièrement basée sur ce type de support, il vous en coûtera 11 de plus.

Voici les fichiers à récupérer sur **http://ftp.fr.debian.org/debian/dists/**. Sur ce serveur, vous trouverez trois répertoires déterminant la stabilité des packages et de la distribution en général : stable (*potato*), en test (*woody*) et non stable (*sid*). Nous vous conseillons une distribution stable vous assurant le moins de problème à l'utilisation. Les fichiers à récupérer se trouvent dans **stable/main/disks-i386/current/images-1.44** :

- *rescue.bin* est la disquette de démarrage.
- *root.bin* est le système de fichier root contenant toute la procédure d'installation.
- *drivers-1.bin* à *drivers-4.bin* sont les disquettes qui contiennent tous les modules vous permettant de supporter le matériel de votre machine (carte réseau en particulier).
- *base-1.bin* à *base-11.bin* sont les disquettes qui composent le système de base. Si votre connexion à Internet est directement utilisable par la procédure d'installation, vous n'avez pas besoin de ces disquettes. Le système de base sera récupéré sur le serveur officiel Debian automatiquement.

Une fois les fichiers récupérés, vous devrez en faire des disquettes. Sous Linux, utilisez la commande :
 # dd if=disque.bin of=/dev/fd0

où *disque.bin* est le fichier récupéré sur le serveur Debian et */dev/fd0* le périphérique lecteur de disquette. Attention, vous devez être root pour accéder au périphérique de cette manière.
Sous Windows/DOS, vous devrez récupérer l'exécutable *stable/main/disks-i386/current/dosutils/rawrite2.exe*. Vous placerez ensuite ce fichier avec les images de disquettes dans un répertoire et lancerez *rawrite2*. Vous serez invité à entrer successivement un nom de fichier image et un lecteur de disquette à utiliser.

Une fois toutes les disquettes générées, placez la disquette *rescue* dans le lecteur de la machine devant être le serveur HTTP et démarrez. L'ordinateur va alors charger le kernel Linux et initialiser le système. Vous serez ensuite invité à entrer la disquette root. Après chargement, la procédure d'installation démarrera avec le choix d'organisation du clavier (*latin-1* pour la France).

Vous n'aurez plus qu'à suivre les indications et la procédure pour installer le système. En ce qui concerne les partitions du système GNU/Linux, il est recommandé que la première partition du disque soit de petite taille et soit montée en tant que */boot*. Celle-ci est destinée à stocker le kernel et doit donc être accessible par le gestionnaire de *boot* (*bootloader*).
Une autre partition spécifique est recommandée : la partition de *swap*. C'est elle qui fera office de mémoire si la RAM de votre système s'avérait insuffisante. Définissez une taille de partition égale au double de votre mémoire vive disponible.
Enfin, le reste du disque peut être alloué à la racine (*/*) du système de fichiers.

Lors de la procédure d'installation, une étape consiste à charger des modules permettant d'ajouter le support de certain matériel et en particulier des interfaces réseau. Il vous sera alors demandé de placer successivement les 4 disquettes drivers dans le lecteur. Chargez ensuite les modules concernant votre interface réseau.
Note : Dans le cas de l'utilisation d'une carte ISA de type ne2000 ou compatible, vous devrez charger le module *8390.o* avant *ne.o*. Il est fort probable que vous possédiez ce genre de carte très courante sur les petites configurations et l'absence du module 8390 est une cause fréquente d'erreur.

Viendra ensuite la configuration réseau. Les choix que vous allez faire ici perdureront dans la configuration du système installé. Vous pourrez, bien sûr, changer cette configuration par la suite en éditant les fichiers adéquats. Si vous utilisez une connexion à Internet par le câble, branchez directement le modem câble sur l'interface réseau et répondez affirmativement en ce qui concerne l'utilisation d'un client DHCP.

Pour ce qui est de l'ADSL, vous devrez préciser l'adresse IP de la machine faisant office de passerelle pour la connexion à Internet.

Enfin, si vous ne pouvez avoir accès à Internet, choisissez une installation par disquette pour le système de base. Vous pourrez ensuite, après le redémarrage, configurer l'accès réseau et au besoin installer les packages depuis une disquette avec *dpkg*.

Le système de base, une fois configuré pour l'accès à Internet, est suffisant pour suivre les indications du magazine. Nous expliquerons comment installer les packages de chaque logiciel traité.

La connexion ADSL

De plus en plus populaire, l'ADSL remplace petit à petit les autres modes de connexion à Internet comme le câble ou même les liaisons louées. L'ADSL utilise comme support les lignes téléphoniques existantes en faisant usage de plages de fréquence non utilisées par les communications téléphoniques classiques. De ce fait, une simple prise téléphonique peut devenir votre point d'accès à Internet à haut débit.

Les deux principaux avantages en faveur de l'ADSL sont l'absence d'installation particulière (une prise de téléphone fixe suffit) et, bien sûr, le prix. Il existe plusieurs fournisseurs d'accès à Internet proposant une connectivité ADSL, mais il n'existe qu'un seul fournisseur pour cette connectivité : Orange/FT. En effet, il vaut voir la connexion ADSL en deux parties :

• un réseau physique mis en place par Orange et connu sous le nom de Netissimo. Avec Netissimo, vous n'avez pas accès à Internet, mais uniquement au WAN ADSL via une plaque ADSL locale.

• un service d'accès à Internet fourni par un FAI, le plus connu étant Wanadoo. Mais il faut savoir qu'un certain nombre de fournisseurs offre des services plus adaptés à l'installation d'un serveur Web. Sans vouloir faire de publicité, il semble que Nerim soit de plus en plus populaire pour l'installation de ce type de système. Le gros avantage est la fourniture d'une adresse IP fixe sur simple demande. Ce qui s'avère fort agréable pour un serveur HTTP ou autre.

En ce qui concerne les tarifs de ce genre de prestations, comptez environ 400 FF en tout et pour tout par mois. Ce montant comprend l'abonnement au Netissimo et le service du fournisseur d'accès. Notez qu'il existe également une offre spécifique à Wanadoo regroupant les deux services dans un pack comprenant un modem ADSL.

Les modems

L'ADSL est accessible par une ligne téléphonique sans gêner les communications classiques. Vous pouvez donc parfaitement être connecté en permanence à Internet tout en recevant et envoyant des appels téléphoniques. C'est un matériel spécifique, branché sur la ligne, qui se chargera d'interfacer votre ordinateur au réseau ADSL : le modem.
Il existe deux grandes familles de modems, différenciées par leur connectivité :

• USB : Celui-ci se connecte à un port USB de votre machine. Je tiens cependant à préciser plusieurs points d'importance concernant ce type de modem. Bien qu'il soit fourni dans le pack le plus économique de Wanadoo, il faut prendre en compte certains éléments comme le fait que l'USB n'est pas disponible sur les petites configurations. C'est un point capital puisque nous partons du principe ici que nous allons réutiliser un PC de petite capacité (486 ou Pentium) en guise de serveur Web. Il est cependant possible d'ajouter dans la machine un contrôleur USB sous la forme d'une carte PCI. Un autre problème persiste, celui des performances. En effet, un contrôleur USB est directement géré par le processeur de la machine.

Une consommation de ressources non négligeable en découle. C'est un avis strictement personnel que je vais vous exposer, mais je pense préférable d'investir une centaine de francs supplémentaires afin de disposer d'un modem de type Ethernet. Je parle de coût supplémentaire car ce type de périphérique n'est disponible qu'avec les offres Netissimo et non le pack "tout compris".

• Ethernet : Ce type de modem se connecte à une interface réseau Ethernet à l'aide d'un connecteur de type RJ45. Il convertit ainsi les signaux Ethernet en signaux ADSL. Ce type de modem est nettement plus facile à installer sur une configuration GNU/Linux et ne nécessite pas de port spécifique sur la machine.

Installation USB

L'un des pilotes les plus utilisés pour les modems ALCATEL SpeedTouch USB est le fruit d'un développeur français, Benoît Papillaut. Celui-ci a développé son propre pilote pour ce modem et il est reconnu par tous qu'il s'avère d'une grande qualité. Depuis peu, il est même possible d'installer ce pilote

sans avoir à recompiler le kernel Linux (si ce n'est pour le support USB).

Les informations qui vont suivre sont directement tirées de la page de Benoît et n'ont malheureusement pas pu être vérifiées par nos soins. Il ne devrait cependant pas y avoir de problème particulier puisque les pages de Benoît sont régulièrement mises à jour.

La première étape consiste à vérifier la présence du support USB dans votre kernel et par-là même la correcte détection du modem en tant que périphérique attaché au système :

```
# cat /proc/bus/usb/devices
[...]
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr=  0mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=00
Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   8 Ivl=255ms
T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=01 Dev#=  4 Spd=12  MxCh=
0
D:  Ver= 1.10 Cls=ff(vend.) Sub=00 Prot=00 MxPS= 8 #Cfgs=  1
P:  Vendor=06b9 ProdID=4061 Rev= 0.00
S:  Manufacturer=ALCATEL
S:  Product=Speed Touch USB
S:  SerialNumber=0090D013AAB8
[...]
```

Ceci fait, vous devrez vérifier un certain nombre d'autres éléments :

- Assurez-vous d'avoir le support PPP actif dans votre kernel et le démon `pppd` présent.
- Assurez-vous d'avoir également le support HDLC.
- Récupérez les pilotes officiels Alcatel (**http://www.alcatel.com/consumer/dsl/supuser.htm**), mais ne les installez pas ! Vous devez simplement récupérer un fichier de ces pilotes : `mgmt`, `mgmt.o` ou `alcaudsl.sys`.
- Enfin, récupérez les pilotes de Benoît sur **http://benoit.papillault.free.fr/speedtouch/index.php3**.

Avant de démarrer la compilation du nouveau pilote, vous aurez besoin de l'environnement de développement GNU Gcc et des sources de votre kernel actuel (placés dans `/usr/src`).
Ceci fait, vous pourrez décompresser l'archive du pilote et dans le répertoire ainsi créé, utiliser les commandes suivantes pour compiler le pilote et l'installer :

```
# make
# make install
```

Il vous faudra ensuite configurer les fichiers permettant la connexion. Vous devez disposer de votre identifiant, de votre mot de passe et du couple `vpi.vci` (normalement 8.35) fournis par le FAI auquel vous vous êtes adressé.

`/etc/ppp/peers/adsl` :

```
debug
kdebug 1
noipdefault
defaultroute
pty "/usr/local/bin/pppoa2 -vpi votre_vpi -vci votre_vci"
sync
user "votre login"
novjccomp
noaccomp
nopcomp
nomagic
noccp
asyncmap 0
usepeerdns
holdoff 1
persist
maxfail 0
```

`/etc/ppp/chap-secrets` :

```
# Secrets for authentication using CHAP
# client                server  secret          IP addresses
votre_login *       votre_mot_de_passe
```

Ceci fait, vous pourrez lancer la connexion avec :

```
# /usr/local/bin/modem_run -f /chemin/vers/mgmt.o -m
# pppd call adsl
```

La première ligne charge le microcode dans le modem USB (`mgmt.o` doit être remplacé par le nom de fichier provenant des pilotes officiels d'Alcatel). La seconde ligne lance la connexion ADSL.

Les indications que nous avons données ici sont très sommaires ; en cas de problème, reportez-vous directement aux pages de Benoît où vous trouverez des explications plus détaillées ainsi qu'une FAQ.

Installation Ethernet

En achetant une offre Netissimo + FAI, vous aurez à votre disposition un modem Ethernet. Il n'est pas nécessaire alors d'utiliser un quelconque pilote supplémentaire. Vous devrez simplement configurer votre interface Ethernet normalement puis installer un package Debian supplémentaire :

```
# apt-get install pppoe
```

Le package `pppoe` contient le support *PPP over*

Ethernet développé par Roaring Penguin. Cette commande vous installe plusieurs fichiers de configuration et quelques scripts permettant de faciliter la procédure de connexion ADSL :

`/etc/ppp/pppoe.conf` contient toutes les informations nécessaires au fonctionnement de PPPoe. Vous n'avez, en principe, qu'une seule ligne à modifier dans ce fichier :

```
USER=netissimo@netissimo
```

`USER` prend en argument l'identifiant de connexion fourni par votre FAI. Celui-ci est utilisé ensuite pour chercher les informations d'authentification dans le fichier `pap-secrets` dans le même répertoire :

```
# Secrets for authentication using PAP
# client        server  secret          IP addresses
netissimo@netissimo     *       netissimo       *
```

Enfin, le fichier `options.pppoe` contient les derniers éléments de configuration pour le démon `pppd` :

```
lock
noipdefault
holdoff 30
lcp-echo-failure 4
lcp-echo-interval 30
mtu 1492
mru 1492
receive-all
noauth
noaccomp
```

Les deux lignes très importantes dans ce fichier concernent les valeurs de `mtu` et `mru` (taille de *frame*). En effet, un réseau Ethernet classique utilise une valeur de 1500, alors que le réseau ADSL en utilise d'autres. Il nous faut donc préciser une taille inférieure afin de s'assurer que tous les paquets arriveront bien à passer sur l'Ethernet.

Une fois tout cela correctement configuré, vous n'aurez plus qu'à lancer la connexion avec :

```
# /usr/sbin/adsl-connect
```

N'oubliez pas cependant de configurer les serveurs DNS dans le fichier `/etc/resolv.conf`. Ces valeurs vous ont été normalement fournies en même temps que vos paramètres de connexion.

Vérification

Quel que soit le modem ou les pilotes que vous utiliserez, si la connexion réussit, vous devez trouver une nouvelle interface réseau sur votre système :

```
# ifconfig
[...]
ppp0      Link encap:Point-to-Point Protocol
          inet addr:62.212.96.XXX  P-t-P:194.206.78.3
          Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1492
          Metric:1
          RX packets:18253 errors:0 dropped:0 overruns:0
          frame:0
          TX packets:20694 errors:0 dropped:0 overruns:0 car-
          rier:0
          collisions:0 txqueuelen:10
[...]
```

Vous pouvez également inspecter les journaux système afin de détecter un éventuel problème et ainsi corriger les erreurs qui se trouvent dans les fichiers de configuration :

```
pppd[9951]: pppd 2.3.10 started by root, uid 0
pppd[9951]: Using interface ppp0
pppd[9951]: Connect: ppp0 <--> /dev/pts/1
pppoe[9953]: PADS: Service-Name: ''
pppoe[9953]: PPP session is 4586
pppd[9951]: sent [LCP ConfNak id=0x59 <auth pap>]
pppd[9951]: user="XXXXXXXXX@net1.nerim.fsa" password=<hidden>]
pppd[9951]: <mru 1492> <magic 0x21c4feb8>]
pppd[9951]: <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns3 0.0.0.0>]
pppd[9951]: local  IP address 62.212.96.249
pppd[9951]: remote IP address 194.206.78.3
pppd[9951]: primary   DNS address 62.4.16.70
pppd[9951]: secondary DNS address 62.4.17.109
```

Le présent article ne se veut pas exhaustif (loin de là), il décrit une procédure de connexion ADSL qui fonctionne immédiatement et sans problème. Il peut malheureusement arriver que vous rencontriez un problème spécifique. La réaction à avoir dans ce cas est de visiter les pages concernant les connexions ADSL sur Internet (voir Liens) et les groupes de discussion où vous trouverez une âme charitable pour vous aider...

Liens

RP-PPPoe :
http://www.roaringpenguin.com/pppoe/

Le driver GPL pour le modem ADSL SpeedTouch USB :
http://benoit.papillault.free.fr/speedtouch/index.php3

Groupes Usenet :
fr.comp.os.linux.configuration
fr.reseaux.telecoms.adsl

Le câble : DHCP

Un fournisseur Internet de type "câble" fournit aussi bien une diffusion des émissions télé qu'une connectivité Internet. Le réseau formé par le support de communication qu'est le câble est un WAN (Wide Area Network). La connexion de l'interface réseau de votre ordinateur sur ce réseau se fait par l'intermédiaire d'un modem qui transformera le signal câble en signal Ethernet. La configuration de cette interface se fera par DHCP.

Le protocole DHCP (*Dynamic Host Configuration Protocol*) permet une configuration automatique des interfaces réseau. Entendez par là que votre interface obtiendra automatiquement et dynamiquement une configuration IP. Ce protocole est normalement utilisé de pair avec BOOTP (*Bootstrap Protocol*). Ces deux protocoles permettent historiquement la configuration et l'installation de machines *diskless*, c'est-à-dire ne comprenant aucun système de fichiers statiques (disque dur, CD ou disquette).

Dans le cas du réseau câble, seul DHCP est utilisé pour fournir automatiquement une configuration au client du FAI. DHCP est couvert par les informations techniques disponibles via la RFC 1541. Si vous désirez absolument tout connaître sur ce protocole, c'est ce document qu'il faudra consulter. La caractéristique importante à retenir ici est que votre modem câble ne doit être vu que comme un transformateur de signaux. C'est votre carte Ethernet qui sera directement en relation avec l'ensemble du WAN du fournisseur d'accès.

Si vous disposez d'une connexion Internet via le câble, vous êtes un client DHCP. Le FAI dispose pour sa part d'une machine faisant office de serveur DHCP. De manière générale, les FAI ne configurent pas l'attribution d'une adresse IP de manière statique. En clair, l'adresse IP est fournie au client en fonction des disponibilités du FAI. En cas de déconnexion du réseau (arrêt du modem, arrêt de la machine cliente, coupure physique de la connexion ou relance du client DHCP), vous avez de fortes chances que votre adresse IP (et même votre configuration TCP/IP en général) change.

Cette caractéristique est très importante dans le cadre de la mise en oeuvre d'un serveur Web. Il ne faut en aucun cas que du fait d'un changement d'adresse IP, votre serveur Web devienne inaccessible. Nous traiterons de ce paramètre un peu plus loin dans cet article.

Installation du client DHCP

L'installation d'un client DHCP est d'une facilité déconcertante. Il vous suffira en effet d'installer le package `dhcpcd`. Il s'agit du démon client pour le protocole DHCP.
Une fois l'archive installée (via une autre machine sur le réseau local ou un package installé avec `dpkg -i`), un nouveau service aura été installé dans le système d'init. Vous trouverez en effet un fichier `/etc/init.d/dhcpcd` permettant de lancer ou de tuer le client DHCP.

En principe, celui-ci ne devra pas être lancé manuellement. Sur une distribution Debian, ce script d'init est lancé via la configuration du réseau. De plus, si vous avez procédé à une installation Debian via Internet, la procédure d'installation vous aura déjà demandé si vous souhaitez obtenir une adresse IP de la part d'un serveur DHCP. Si tel est le cas, vous n'aurez pas besoin d'installer un quelconque package supplémentaire. Le système étant déjà configuré, toute manipulation supplémentaire devient inutile.

Si, en revanche, vous avez installé votre distribution depuis une autre source (CD-ROM, autre machine du réseau local en NFS ou encore via disquettes) aucune configuration spécifique n'est nécessaire puisque le script de configuration du package `dhcpcd` a dû normalement modifier votre configuration réseau.

IP dynamique

Nous l'avons dit plus haut, les fournisseurs d'accès Internet par le câble attribuent normalement les adresses IP des clients de manière dynamique. C'est également le cas de certains fournisseurs d'accès ADSL chez lesquels l'obtention d'une adresse IP

fixe est impossible avec un abonnement standard. Vous ne pouvez donc pas utiliser une entrée "classique" dans un DNS public (IN A) puisque ce type d'enregistrement est statique et qu'une modification de l'adresse IP mettra plus de 24 heures à être propagée.

Il faut donc ruser et trouver un moyen de prendre en compte dynamiquement le changement d'adresse et ce, à la même vitesse que le changement opéré par le serveur DHCP.
L'astuce consiste donc à ne pas associer directement votre nom de domaine nouvellement acquis avec une adresse IP susceptible de changer. Il faut associer votre IP à un nom de machine sur un réseau spécifique (et donc intégrant un serveur DNS interne), puis créer un alias (CNAME) de votre nom de domaine vers ce nom de machine sur le réseau en question. Ainsi, les changements dans le DNS interne du réseau pourront être appliqués immédiatement (il n'est pas nécessaire de les propager à travers tous les serveurs DNS de la planète).

La théorie peut paraître quelque peu complexe mais la pratique clarifie immédiatement les choses. Le principal fournisseur de ce type de service est **dyndns.org** (alias **homeip.net**). Ne considérez pas cela comme du bricolage, **dyndns.org** traite actuellement plus de 128 000 domaines de cette manière. En ouvrant un compte gratuit sur **dyndns.org**, vous spécifierez plusieurs paramètres comme vos coordonnées personnelles, mais aussi et surtout votre adresse IP actuelle. Ceci aura pour effet (avec un petit délai pour l'ouverture du compte) de créer immédiatement une nouvelle machine sur le réseau DynDNS avec la syntaxe : *identifiant.dyndns.org* ou encore *identifiant.homeip.net*. Dès lors, votre machine sera accessible directement via ce nom.

Ce principe est très différent de la mise en œuvre d'un DNS public puisqu'il ne s'agit pas d'un nouveau domaine mais d'une nouvelle machine dans un domaine existant. Le seul DNS concerné est donc celui de DynDNS qui est régulièrement et directement mis à jour en cas de changement.

Ainsi, si votre adresse IP attribuée dynamiquement par le fournisseur d'accès à Internet venait à changer, il vous suffirait de répercuter ces changements dans les paramètres de votre compte DynDNS. Bien sûr, le faire manuellement n'aurait qu'un intérêt limité. C'est pourquoi des clients DynDNS ont été développés aussi bien par des membres du réseau que sous forme de contributions. Un client DynDNS est un utilitaire qui va, à intervalle régulier, vérifier l'adresse IP de votre connexion Internet. En cas de changement, l'utilitaire en question contactera à votre place le serveur DynDNS et procédera aux modifications adéquates. Il existe plusieurs clients pour chaque système (GNU/Linux, Windows, MacOS, etc.). Vous trouverez une liste complète des clients DynDNS pour les systèmes Unix/Linux sur **http://support.dyndns.org /dyndns/clients/unix.shtml**.

En plus d'une mise à jour de votre compte DynDNS en cas de changement d'adresse IP, vous devez prendre en compte le fait qu'en l'absence d'activité, le système DynDNS supprimera votre compte. En clair, sans changement de votre part ou de réactualisation du compte sans changement d'adresse, DynDNS supprimera le compte qu'il estimera alors sans intérêt. En effet, pour ne pas engorger le système avec des comptes inutiles, cette vérification est nécessaire. Encore une fois, les utilitaires clients sont conçus pour pallier le problème, et si dans l'éventualité où votre adresse ne change pas durant une trop longue période, le client accèdera à votre compte pour éviter la suppression.

DynDNS ne recommande aucun client en particulier. Cependant, l'un d'entre eux est largement utilisé : *ez-ipupdate*. Vous pourrez vous procurer celui-ci sur **http://gusnet.cx/proj/ez-ipupdate**. A l'heure où nous écrivons ces lignes, la dernière version en date est la 3.0.5. L'auteur de cette application tient très régulièrement son logiciel à jour en apportant les modifications nécessaires en cas de mise à jour ou de changement sur les pages du site officiel.

Une fois les sources du client récupérées, suivez les indications fournies dans le fichier *README* de l'archive. Vous obtiendrez ainsi un binaire (exécutable), `ez-ipudate`. Vous devrez ensuite créer le fichier de configuration comportant toutes les indications nécessaires à la mise à jour automatique de votre compte DynDNS. Voici un exemple de fichier :

```
user=utilisateur:mot_de_passe
interface=eth0
daemon
host=identifiant.homeip.net
service-type=dyndns
```

ez-ipupdate ne limite pas ses fonctionnalités à DynDNS, vous devrez donc préciser qu'il s'agit de ce type de compte avec le paramètre `service-type`. L'option `user=` vous permet d'entrer votre identifiant et votre mot de passe pour votre compte séparé d'un double point (:). `interface` vous permet de spécifier l'interface Ethernet à surveiller en cas de changement. Ici, il s'agit de la première interface (`eth0`), mais votre configuration est peut-être différente. `host`, enfin, est le nom de votre machine sur le réseau DynDNS. Le paramètre `deamon` spécifie à *ez-ipupdate* qu'il doit fonctionner en tant de démon et donc basculer directement en arrière-plan en vous rendant la main. Vous devez cependant pouvoir voir dans le journal système le bon fonctionnement du démon avec des informations comme :

```
ez-ipupdate Version 3.0.3, Copyright (C) 1998-2000
Angus Mackay.
ez-ipupdate started for interface eth0 using ser-
ver members.dyndns.org
successful update for eth0->213.166.192.67
```

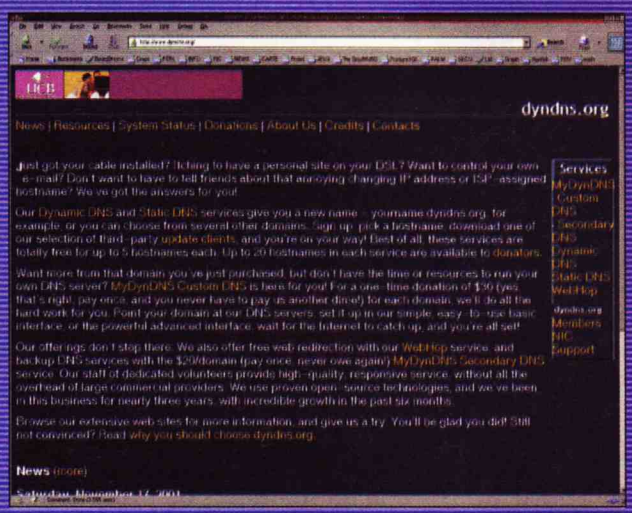
Le démon vous informe qu'il a accédé au serveur DynDNS (**members.dyndns.org**) et qu'il a mis à jour votre compte avec l'adresse IP de votre interface `eth0` avec succès.

Note à propos des abus : DynDNS est un service gratuit permettant à n'importe quel utilisateur possédant un connexion permanente à Internet de créer son propre site Web (ou autre). En dehors du fait que DynDNS accepte, bien sûr, les dons en tout genre, il y a une règle importante à respecter. Certains clients permettent de faire des mises à jour des comptes à intervalles réguliers. Suite à des problèmes techniques provenant d'utilisateurs ayant configuré leur client pour des mises à jour trop fréquentes (toutes les 5 minutes par exemple), des règles ont été définies. Une mise à jour du compte qui ne sera pas considérée comme un abus devra être justifiée par un changement d'adresse IP ou une mise à jour avec une adresse IP identique après le délai de 28 jours d'inactivité. Tout changement en dehors de ces critères sera considéré comme un abus et le compte sera bloqué durant une période de 7 jours. Veuillez donc configurer correctement votre client si celui-ci permet de définir un intervalle de mise à jour du compte.

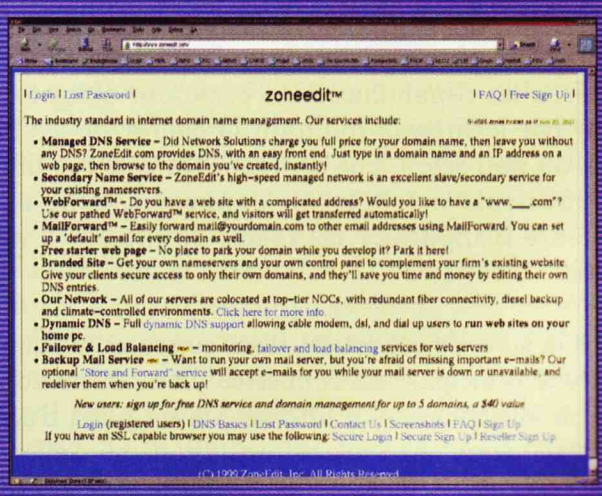
Arrivé à ce point, vous possédez déjà une identité sur Internet avec un nom de machine faisant partie du réseau DynDNS. Il ne reste plus maintenant qu'à créer un alias, autrement dit un CNAME, entre le nom de domaine que vous avez acquis et le nom de machine sur DynDNS. Attention, sur un DNS public comme ZoneEdit, veillez à configurer un CNAME et non une redirection. Un CNAME vous permettra de gérer les hôtes virtuels avec Apache (serveur HTTP). De ce fait, vous pourrez déposer plusieurs noms de domaine étant des alias de la même machine sur DynDNS et votre serveur Web proposera un contenu différent pour chaque site.

Liens

DynDNS :
http://www.dyndns.org



ZoneEdit :
http://www.zoneedit.com



Le filtrage de paquets sous Linux : Mise en place d'un firewall personnel

Un des meilleurs moyens d'éviter les tentatives de piratage reste encore le filtrage, dès l'entrée du réseau, de tout ce qui n'est pas censé y pénétrer. Cette tâche est généralement remplie par une machine qui joue le rôle de "firewall" (pare-feu en français).
Dans cet article, nous allons détailler les bases indispensables à la mise en place et la configuration d'un tel système.
En particulier, nous examinerons de plus près la mise en place d'un firewall "personnel" pour protéger un mini-réseau connecté par une liaison intermittente avec une seule adresse IP (modem, ADSL, câble et quasiment tous les accès Internet non "professionnels").

Une seule machine ou plusieurs ?

Le dispositif de filtrage peut être considéré comme un filet qui va stopper un certain nombre de paquets indésirables.

Afin de pouvoir filtrer efficacement les paquets, le dispositif de filtrage doit être physiquement intercalé entre le réseau qu'il protège et le "reste du monde". Concrètement, il y a deux manières de procéder.

Cela se fait avec une machine possédant 2 interfaces réseau (Ethernet la plupart du temps), l'une connectée sur la partie interne du réseau et l'autre sur le routeur qui permet d'accéder à l'extérieur. De cette manière, les communications passeront obligatoirement par le firewall qui aura la charge de les bloquer ou non selon leur contenu. Dans le cas limite où une seule machine est connectée, celle-ci se contente d'appliquer les règles de filtrage à elle-même.

Dans la suite de cet article, nous supposerons que la machine principale est connectée (de manière continue ou intermittente) à Internet via un point d'accès qui n'alloue qu'une seule adresse IP et qui n'effectue aucun filtrage.

Règles de bases du filtrage

Nous devons maintenant déterminer ce que notre filtre de paquets IP va effectivement devoir éliminer ou au contraire accepter de laisser passer.

Il existe deux principales manières de configurer un tel filtre :

• La bonne manière – Par défaut, aucun paquet ne passe, sauf ceux qu'une règle autorise explicitement à passer.

• La mauvaise manière (malheureusement assez souvent employée) – Seuls les paquets explicitement interdits sont stoppés, les autres passent.

L'explication est simple : dans le premier cas, l'oubli d'une règle de filtrage résulte en un service qui ne fonctionne pas comme il devrait, voire pas du tout. On s'en aperçoit alors généralement assez vite et il suffit d'ajouter la règle adéquate pour que tout rentre dans l'ordre.

Dans le deuxième cas, un oubli crée un problème de sécurité potentiel qui risque d'être long et difficile à mettre en évidence, si tant est qu'on le découvre un jour.

Netfilter

Le logiciel de filtrage le plus utilisé sous Linux 2.4 est Netfilter ; il remplace avantageusement ipchains qui était utilisé par la version 2.2 du kernel Linux. Netfilter est composé de deux parties : le support kernel qui doit être compilé dans votre noyau, ainsi que la commande `iptables` qui devrait déjà être disponible sur votre système.

Exemple de mise en place

Un exemple commenté valant mieux qu'un long discours, nous allons dans la suite de cet article décrire l'installation et la mise en place d'un dispositif de filtrage. On suppose que la configuration pour la connexion à Internet proprement dite a déjà été effectuée et est fonctionnelle.

Commencez par vous assurer que votre noyau comporte effectivement le support de Netfilter. Si c'est le cas, les deux lignes suivantes devraient être présentes dans les messages de boot (ceux-ci peuvent être fournis par la commande `dmesg`) :

```
ip_conntrack (4095 buckets, 32760 max)
ip_tables: (c)2000 Netfilter core team
```

Dans le cas contraire, vous devrez recompiler votre noyau après avoir activé le support Netfilter. Les options correspondantes se situent dans le sous-menu "`Network Packet Filtering`" du menu "`Networking Options`". Dans la section "`Netfilter Configuration`", choisissez les options qui vous intéressent. Dans le doute, vous pouvez toutes les sélectionner. Si vous avez plusieurs machines sur votre réseau, pensez en particulier à activer l'option "`Full NAT`" ainsi que la sous-option "`MASQUERADE target support`". Les noms qui précèdent sont ceux utilisés par le noyau 2.4.13, mais ces noms ne devraient pas changer sensiblement dans les versions ultérieures. Il est également préférable d'inclure Netfilter directement dans le noyau et de ne pas utiliser de modules. En effet, si pour une raison ou pour une autre, un des modules de Netfilter venait à manquer ou n'était pas chargé, le filtrage serait inopérant avec tous les risques de sécurité que cela comporte.

Si votre liaison Internet se fait par modem, vous ne devriez avoir besoin que d'une seule carte réseau dont la détection ne devrait pas poser de problème particulier. En revanche, dans le cas où la connexion à Internet se fait directement par Ethernet (via un routeur ou un "modem-câble" par exemple), vous aurez besoin de deux cartes réseaux.

Il est important de noter que le kernel Linux, lors de l'auto-détection du matériel, arrête la recherche des cartes réseaux dès qu'il en a trouvé une. Par défaut, une seule carte sera donc détectée.
Ce problème est facilement contourné en ajoutant la ligne suivante dans le fichier `lilo.conf` :

```
append="ether=0,0,eth1"
```

Nous devons maintenant configurer les interfaces Ethernet. La méthode que nous avons choisie permet d'utiliser la même adresse IP sur les deux cartes, permettant ainsi d'économiser une adresse supplémentaire.

Dans ce qui suit, nous allons supposer que nous disposons du sous-réseau 172.16.6.0/24, c'est-à-dire des adresses allant de 172.16.6.1 à 172.16.6.254 incluses. Dans notre exemple, la passerelle de connexion aura pour adresse 172.16.6.1. D'autre part, l'interface `eth1` sera connectée au hub/switch reliant entre elles les machines du réseau local.

L'interface interne sera donc initialement configurée avec les paramètres suivants :

```
address   : 172.16.6.1
netmask   : 255.255.255.0
network   : 172.16.6.0
broadcast : 172.16.6.255
```

Dans la mesure où l'on ne dispose que d'une seule adresse IP et que l'on désire connecter plusieurs machines, nous allons être forcés d'utiliser un subterfuge nommé "`IP Masquerading`". Celui-ci permet à la machine qui fait office de passerelle d'intercepter tous les paquets provenant du réseau interne à destination de l'extérieur et remplace l'adresse source contenue dans l'en-tête du paquet par la sienne. Lorsqu'une réponse est reçue, l'opération est faite en sens inverse et le paquet est acheminé à la machine du réseau local qui a émis la requête initiale. Ce système offre également comme avantage d'offrir directement un semblant de firewall aux machines du réseau local, car seules celles-ci peuvent initier une connexion vers l'extérieur. En revanche, la machine qui possède l'adresse IP publique doit quant à elle être protégée avec les règles adéquates.

Pour commencer, voici le script qui permet de mettre le masquerading en place.

net-config.sh : Script de configuration réseau

```
#!/bin/sh
# Décommenter la ligne suivante pour afficher les
commandes lors de leur exécution
# set -x
# Dans la ligne suivante remplacez éventuellement
'ppp0' par 'eth0' si la
# connexion à Internet se fait par une carte
Ethernet.
iptables -t nat -A POSTROUTING -o ppp0 -j
MASQUERADE
# On active le "forwarding" pour permettre aux
paquets arrivant sur
# une interface d'être routés vers l'autre et
vice-versa.
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Si vous êtes réellement certain de ne pas vouloir de filtrage, vous pouvez vous contenter de ce script. En revanche, si vous désirez une protection pour la machine qui fait office de passerelle voici comment faire.

Il s'agit maintenant de mettre le filtrage en place grâce à Netfilter.

Netfilter permet d'agir directement sur le cheminement des paquets. Dans la configuration de base, les paquets passent par trois chaînes de règles :

- INPUT : par laquelle passent tous les paquets entrant par une interface ;
- FORWARD : par laquelle passent tous les paquets qui sont transmis d'une interface à une autre ;
- OUTPUT : par laquelle passent les paquets avant de sortir par une interface.

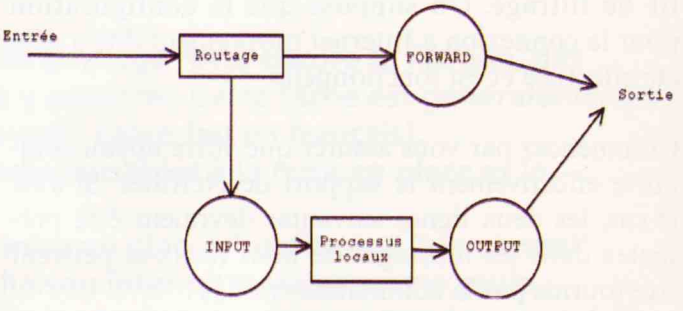
La commande `iptables` permet d'ajouter, de modifier ou de retirer des règles dans chacune de ses chaînes pour modifier le comportement du filtrage.

De plus, chaque chaîne possède une politique (*policy*) par défaut, c'est-à-dire le comportement à suivre lorsque aucune règle de la chaîne n'a correspondu au paquet.

Les quatre comportements les plus courants sont :

- ACCEPT : On laisse passer le paquet.
- REJECT : On rejette le paquet et on envoie le paquet d'erreur associé (*ICMP Port Unreachable, TCP RESET*, selon les cas).
- LOG : Enregistre une notification du paquet dans le "syslog".
- DROP : Le paquet est tout simplement ignoré et aucune réponse n'est envoyée.

Cheminement des paquets dans les chaînes standards



Voici les principales options d'`iptables` permettant de manipuler des chaînes entières. Nous reviendrons ensuite un peu plus en détail sur chacune d'entre elles :

- N : Création d'une nouvelle chaîne.
- X : Suppression d'une chaîne vide.
- P : Changement de la politique par défaut d'une chaîne.
- L : Liste les règles d'une chaîne.
- F : Élimine toutes les règles d'une chaîne.
- Z : Remet à zéro les compteurs d'octets et de paquets ayant traversé la chaîne.

En ce qui concerne les modifications de chaînes, les commandes suivantes sont disponibles :

- A : Ajoute une règle à la fin d'une chaîne.
- I : Insère une nouvelle règle à une position donnée dans une chaîne.

• R : Remplace une règle donnée dans une chaîne.

• D : Efface une règle dans une chaîne, soit par son numéro d'ordre, soit en décrivant la règle.

Voyons tout de suite un petit exemple pratique : nous allons interdire les réponses PING (c'est-à-dire les paquets ICMP de type "echo-reply") venant d'une machine donnée.

Commençons par nous assurer qu'il est effectivement possible, pour le moment, de "pinger" la machine en question :

```
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255
time=0.6 ms
--- 172.16.6.74 ping statistics ---
1 packets transmitted, 1 packets received, 0% pac-
ket loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

Maintenant, nous allons ajouter une règle dans la chaîne INPUT qui va intercepter les paquets ayant pour source la machine 172.16.6.74 (`-s 172.16.6.74`), de type ICMP-Reply (`-p icmp --icmp-type echo-reply`). Ces paquets seront tout simplement ignorés (*-j DROP*).

```
# iptables -A INPUT -s 172.16.6.74 -p icmp --icmp-
type echo-reply -j DROP
```

Maintenant, essayons à nouveau un PING vers cette machine :

```
# ping -c 3 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
--- 172.16.6.74 ping statistics ---
3 packets transmitted, 0 packets received, 100%
packet loss
```

Comme nous pouvions nous y attendre, cette fois les réponses au PING ne passent plus. D'autre part, nous pouvons vérifier que les trois réponses ont bien été bloquées (3 paquets, pour un total de 252 octets) :

```
# iptables -L INPUT -v
Chain INPUT (policy ACCEPT 604K packets, 482M bytes)
 pkts bytes target     prot opt in     out
source               destination
    3   252 DROP       icmp --  any    any
172.16.6.74          anywhere
```

Pour revenir à la situation initiale, il nous suffit de demander la suppression de la première règle de la chaîne INPUT :

```
# iptables -D INPUT 1
```

Et maintenant, le PING devrait à nouveau fonctionner correctement :

```
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255
time=0.6 ms
--- 172.16.6.74 ping statistics ---
1 packets transmitted, 1 packets received, 0% pac-
ket loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
#
```

Ça marche !

En plus des trois chaînes pré-existantes (qu'il est d'ailleurs impossible de supprimer), il est également possible de créer d'autres chaînes et d'y faire passer une certaine partie du trafic. Ceci est très utile pour, par exemple, éviter de dupliquer des règles communes entre plusieurs chaînes.

Attachons-nous maintenant à mettre en place les règles nécessaires pour un firewall minimaliste. Celui-ci laissera passer les services *ssh*, *domain* (DNS), *http* et *smtp* à l'exception de tous les autres.

Pour simplifier les choses, les commandes pour mettre les règles en place sont enregistrées dans un script shell pour rendre la configuration plus pratique. Le script commence par enlever toute la configuration actuelle du filtrage avant de mettre la nouvelle en place. Cette petite astuce permet au script d'être "idempotent", c'est-à-dire que l'on peut le relancer même si la configuration est déjà active sans risque d'avoir de règle en double.

rc.firewall

```
#!/bin/sh
# Effacement de toutes les règles
iptables -F
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
# On active le masquerading
iptables -t nat -A POSTROUTING -o ppp0 -j
MASQUERADE
# On construit une chaîne par direction.
```

```
# bad = eth0 pour ppp0 (extérieur)
# dmz = eth1 (réseau local)
iptables -X bad-dmz
iptables -N bad-dmz
iptables -X dmz-bad
iptables -N dmz-bad
iptables -X icmp-acc
iptables -N icmp-acc
iptables -X log-and-drop
iptables -N log-and-drop
# Chaîne spéciale destinée a logguer les paquets
avant de les bloquer
iptables -A log-and-drop -j LOG --log-prefix "drop "
iptables -A log-and-drop -j DROP
# On élimine les paquets ayant tous les flags TCP
activés ainsi que ceux
# avec aucun flag activé (souvent utilisé par les
scans de Nmap)
iptables -A FORWARD -p tcp --tcp-flags ALL ALL -j
log-and-drop
iptables -A FORWARD -p tcp --tcp-flags ALL NONE -j
log-and-drop
# On bloque les paquets provenant des classes
d'adresses réservées
# ainsi que le multicast
iptables -A FORWARD -i eth+ -s 224.0.0.0/4 -j
log-and-drop
iptables -A FORWARD -i eth+ -s 192.168.0.0/16 -j
log-and-drop
iptables -A FORWARD -i eth+ -s 10.0.0.0/8 -j
log-and-drop
# On accepte les paquets appartenants à une
connexion déjà établie
iptables -A FORWARD -m state --state INVALID -j
log-and-drop
iptables -A FORWARD -m state --state RELATED,
ESTABLISHED -j ACCEPT
# Selon le sens d'arrivée des paquets on transmet
à la chaîne correspondante
iptables -A FORWARD -s $DMZ_ADDR -i $DMZ_IFACE -o
$BAD_IFACE -j dmz-bad
iptables -A FORWARD -o $DMZ_IFACE -j bad-dmz
# On ignore tout le reste
iptables -A FORWARD -j log-and-drop
# ICMPs acceptés
iptables -A icmp-acc -p icmp --icmp-type
destination-unreachable -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type
source-quench -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type
time-exceeded -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type
echo-request -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type
echo-reply -j ACCEPT
iptables -A icmp-acc -j log-and-drop
# Chaîne Extérieur -> Intérieur
# On accepte les services mail, DNS, http(s) et
SSH
iptables -A bad-dmz -p tcp --dport smtp -j ACCEPT
iptables -A bad-dmz -p udp --dport domain -j
ACCEPT
iptables -A bad-dmz -p tcp --dport domain -j
ACCEPT
iptables -A bad-dmz -p tcp --dport www -j ACCEPT
iptables -A bad-dmz -p tcp --dport https -j ACCEPT
iptables -A bad-dmz -p tcp --dport ssh -j ACCEPT
iptables -A bad-dmz -p icmp -j icmp-acc
iptables -A bad-dmz -j log-and-drop
# Chaîne Intérieur -> Extérieur
# On accepte les services mail, DNS, http(s) et
telnet
iptables -A dmz-bad -p tcp --dport smtp -j ACCEPT
iptables -A dmz-bad -p tcp --sport smtp -j ACCEPT
iptables -A dmz-bad -p udp --dport domain -j
ACCEPT
iptables -A dmz-bad -p tcp --dport domain -j
ACCEPT
iptables -A dmz-bad -p tcp --dport www -j ACCEPT
iptables -A dmz-bad -p tcp --dport https -j ACCEPT
iptables -A dmz-bad -p tcp --dport telnet -j
ACCEPT
iptables -A dmz-bad -p icmp -j icmp-acc
iptables -A dmz-bad -j log-and-drop
# Chaines pour la machine elle-même
iptables -N bad-if
iptables -N dmz-if
iptables -A INPUT -i $BAD_IFACE -j bad-if
iptables -A INPUT -i $DMZ_IFACE -j dmz-if
# Interface externe
# On accepte uniquement SSH sur la machine
elle-même
iptables -A bad-if -p icmp -j icmp-acc
iptables -A bad-if -p tcp --dport ssh -j ACCEPT
iptables -A bad-if -p tcp --sport ssh -j ACCEPT
ipchains -A bad-if -j log-and-drop
# Interface Interne
iptables -A dmz-if -p icmp -j icmp-acc
iptables -A dmz-if -j ACCEPT
# On active la passerelle uniquement quand toutes
les règles de
# filtrage sont opérationnelles, pas avant !
 echo 1 > /proc/sys/net/ipv4/ip_forward
```

Un dernier point sur la qualité de service. Linux peut modifier le champ ToS ("Type of Service") des paquets et éventuellement modifier sa valeur pour

rendre le paquet plus ou moins prioritaire. Par exemple, la commande suivante modifie les paquets *ssh* sortants afin d'améliorer la réactivité des connexions.

```
iptables -A OUTPUT -t mangle -p tcp --dport ssh -j
TOS --set-tos Minimize-Delay
```

De même, pour les connexions FTP vous pourrez utiliser l'option `--set-tos Maximize-Throughput` pour améliorer le débit au prix d'une possible détérioration de l'interactivité de la session.

Voilà. Maintenant, vous êtes en possession des bases pour mettre en place un filtrage de paquet efficace. Gardez cependant en tête qu'un firewall n'est pas la panacée en matière de sécurité, mais simplement une précaution supplémentaire. La mise en place d'un dispositif de filtrage ne dispense en rien l'utilisation de mots de passe non triviaux, de logiciels équipés des derniers patchs de sécurité, de la mise en place d'un dispositif de détection d'intrusion, etc.

Références

Proxy-ARP Mini-HOWTO :
http://www.linuxdoc.org/HOWTO/mini/Proxy-ARP-Subnet/index.html

Netfilter :
http://netfilter.samba.org/

Network Address Translation HOWTO :
http://netfilter.samba.org/unreliable-guides/NAT-HOWTO/index.html

Utilisateur de GNU/Linux depuis 1993, Vincent Renardias a commencé à s'impliquer activement dans son développement à partir de 1996 : développeur de la distribution Debian, auteur de la traduction française de The GIMP et de l'environnement GNOME, créateur du groupe d'utilisateurs Linux de Marseille (PLUG)...
Actuellement directeur R&D de la société EFB2, il continue à contribuer activement au système GNU/Linux.

Vincent Renardias <vincent@efb2.com>

Apache : votre serveur HTTP

Ce qui est communément appelé un serveur Web est en fait un serveur HTTP. En d'autres termes, il s'agit d'un service sur une machine permettant de répondre aux demandes des navigateurs qui s'y connectent. Le protocole utilisé pour le dialogue est HTTP (*HyperText Transfer Protocol*). L'un des plus connu des serveurs HTTP existants dans le monde Unix est Apache. Bien qu'il existe un grand nombre d'autres serveurs HTTP (libres ou non libres), Apache est depuis un certain temps reconnu comme étant un classique. Voilà pourquoi nous traiterons d'Apache et non d'autres solutions se basant sur Caudium ou Zope.

Après cette petite mise au point, attachons-nous à comprendre exactement ce que fait un serveur HTTP. Pour cela, nous allons tout simplement prendre une transaction typique comme vous en provoquez tous les jours en surfant sur le Web. Les deux parties en présence sont, d'un côté le client (vous et votre navigateur), et de l'autre le serveur attendant les demandes (écoutant).

Le serveur HTTP écoute habituellement un port spécifique sur un réseau TCP/IP : le 80. Le client pour sa part et après son lancement, va utiliser une information pour engager le dialogue avec le serveur. Cette information est l'URL de la page que l'utilisateur souhaite voir. L'URL (*Universal Resource Locator*) est composée de trois éléments tout aussi importants les uns que les autres :

- Une méthode (*http://, ftp://, https://,* etc.). Avec un navigateur Web, la méthode par défaut est *http://* qui correspond à HTTP.

- Un hôte, qui est le nom complet de la machine (*www.mondomaine.com* par exemple).

- Un chemin absolu, qui indique le fichier demandé. Il peut s'agir d'un fichier HTML, bien sûr, mais également de n'importe quel type de fichier directement utilisable par le navigateur (image, fichier texte) ou non (archive, fichier son, autres).

Une URL typique est, par exemple, *http://www.mondomaine.com/index.html*.

Dans la plupart des cas, le serveur Web tentera de définir un fichier par défaut si aucun n'est spécifié par l'utilisateur. Il s'agit d'une directive de configuration du serveur HTTP. Habituellement, une URL incomplète comme *http://www.mondomain.com* se verra ajouter */index.html* en suffixe. Voilà pourquoi on accède généralement à un fichier de départ grâce à un simple nom de machine (machine.domaine.com). Vous pouvez, bien sûr, spécifier une autre valeur de remplacement (index.php par exemple) dans votre fichier de configuration d'Apache. Nous verrons cela un peu plus loin dans le présent article.

Lorsque l'utilisateur désignera une URL à son navigateur, celui-ci va alors contacter le serveur correspondant et lui envoyer une requête HTTP.

La syntaxe est différente en fonction de la version du protocole HTTP utilisé. Avec HTTP 1.0, une requête pourrait être :

```
GET http://www.mondomain.com HTTP/1.0
```

Avec HTTP 1.1 :

```
GET / HTTP/1.1\
Host: www.mondomaine.com
```

Cette requête utilise tout d'abord une méthode HTTP (ici *GET*, mais il existe aussi *POST*, *DELETE* ou encore *CONNECT*). Avec HTTP 1.1, on précise un URI (*Uniform Resource Identifier*), c'est-à-dire une description de l'endroit où se trouve le fichier désiré sur le serveur. / ici correspond à la racine du serveur HTTP (équivalent au fameux *http://www.mondomain.com*).

Voici un exemple de réponse HTTP que pourrait envoyer le serveur :

• la requête :

```
GET /titi.html HTTP/1.1\
Host: egon
```

• la réponse :

```
HTTP/1.1 200 OK
Date: Tue, 20 Nov 2001 10:33:08 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
PHP/4.0.3pl1
Last-Modified: Tue, 20 Nov 2001 10:31:09 GMT
ETag: "1041c-36-3bfa30ed"
Accept-Ranges: bytes
Content-Length: 54
Content-Type: text/html; charset=iso-8859-1

<html>
<body>
<center>coucou</center>
</body>
</html>
```

Le serveur renvoie des informations le concernant (version, distribution, date système), la taille des informations qu'il va envoyer, leur type et enfin, le contenu du fichier désiré. La première réponse du serveur (HTTP/1.1 200 OK) est importante car le code utilisé permet au navigateur client de connaître la réponse du serveur avant l'arrivée des données. Un code 404 par exemple (que vous avez sûrement déjà rencontré sous la forme *404 : Not Found*) informe le navigateur que le fichier demandé n'existe pas sur le serveur. Il existe un grand nombre de codes, en voici quelques-uns :

200 : OK, la requête a été traitée correctement et sans erreur.

203 : PARTIAL INFORMATION, la requête a été traitée mais les informations retournées sont incomplètes.

204 : NO RESPONSE, la requête a été traitée mais le serveur n'a rien à retourner au client.

400 : BAD REQUEST, la requête est incorrecte, le serveur ne peut la traiter.

403 : FORBIDDEN, la requête a été traitée mais le client n'a pas la permission d'accéder aux données.

404 : NOT FOUND, les données (fichier) demandées n'existent pas sur le serveur.

500 : INTERNAL ERROR, le serveur est incapable de traiter la requête en raison d'une erreur interne (rare sous GNU/Linux ;).

501 : NOT IMPLEMENTED, la requête du client ne peut être traitée car le serveur ne possède pas la fonctionnalité nécessaire pour cela.

En principe, vous n'aurez pas besoin de saisir vous-même les requêtes HTTP puisque ce travail est fait directement par votre navigateur. Il est cependant toujours utile de connaître ne serait-ce que vaguement le fonctionnement des protocoles mis en œuvre par les logiciels que vous utilisez. Dernier point important, Apache comme d'autres serveurs HTTP, gère aussi bien HTTP 1.0 que 1.1. Vous n'avez donc pas à avoir d'inquiétude sur la compatibilité avec un éventuel navigateur trop ancien.

Le schéma en **figure 1** vous donne un résumé du fonctionnement et des transactions entre un serveur et un client HTTP. Si vous désirez absolument tout savoir sur HTTP, vous devez consulter les RFC 1945 (HTTP 1.0) et 2616 (HTTP 1.1).

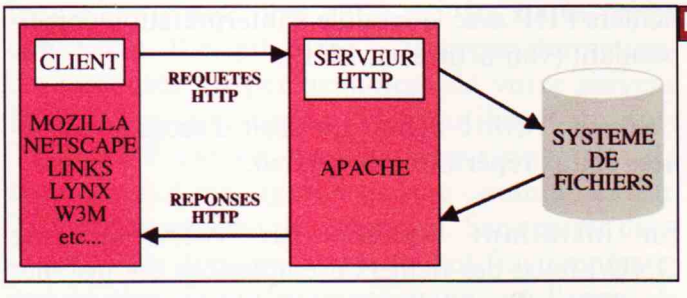


Fig 1

Installation d'Apache

Bienheureux le possesseur d'une distribution Debian car il lui suffira d'utiliser la commande suivante (en tant que root) pour installer Apache :

```
# apt-get install apache
```

Ceci aura pour effet de récupérer tous les packages nécessaires au fonctionnement du serveur HTTP, de démarrer une configuration et de tenter le démarrage du serveur. Selon la configuration réseau de votre distribution, il est possible que le démarrage sur serveur échoue. Ce n'est pas catastrophique puisque, de toute manière, nous allons étudier les principaux éléments du fichier de configuration et, de ce fait, résoudre d'éventuels problèmes.

Si vous utilisez une autre distribution, vous devez avoir à votre disposition un package comprenant dans son nom le mot "apache". Les systèmes de packaging modernes gérant de mieux en mieux les dépendances, vous ne devriez rencontrer aucun problème spécifique qui ne sera corrigé par l'étape de configuration. Seul les emplacements des fichiers d'init et de configuration pourront être différents tout en gardant le même nom (configuration d'Apache).

Configuration

Une configuration Apache repose en premier lieu sur un fichier de configuration appelé `httpd.conf`. Sur une distribution Debian, celui-ci est installé dans le répertoire `/etc/apache`. Ce fichier est le principal fichier de configuration, mais d'autres sont également importants :

- `httpd.conf` regroupe les informations relatives au fonctionnement du serveur HTTP.

- `srm.conf` est le fichier de définition des répertoires et des données du serveur HTTP.

- `mime.types` comprend les informations relatives aux types de données que le serveur HTTP gère. C'est ici par exemple que vous pourrez associer les fichiers PHP avec le module d'interprétation correspondant (voir article sur PHP).

- `access.conf` définit les droit d'accès aux données et aux répertoires du serveur.

En installant Apache sur votre système GNU/Linux, des fichiers exemples ont été installés par défaut. Il est très peu probable que cette configuration vous convienne, mais un certain nombre de paramètres n'auront pas besoin d'être changés pour un serveur Web classique. Entrons dans le vif du sujet en détaillant la nature et l'utilisation des paramètres de chaque fichier.

1) httpd.conf

Ce fichier concerne le fonctionnement et le paramétrage général du serveur Apache. C'est ici que vous configurerez les éléments relatifs au fonctionnement du serveur et non les informations qui y seront placées. Voyons ces paramètres en détail :

• *ServerType standalone* détermine le type de fonctionnement que doit adopter le serveur Apache. Les deux possibilités offertes sont *standalone* (défaut) où le serveur est lancé et écoute le réseau en attente de requêtes HTTP provenant d'un client. Avec ce type de serveur, Apache est constamment en fonctionnement sur la machine et gère lui-même les connexions. L'autre type disponible est *inetd*. Dans ce cas de figure, c'est le démon *inetd* qui attendra les connexions et lancera Apache au besoin. Il n'y a, en temps normal, aucune raison d'adopter un type *inetd*. Je vous conseille donc de laisser la valeur par défaut

• *Port 80* désigne le port TCP/IP à écouter pour les requêtes HTTP. 80 est le port par défaut, utilisé par tous les navigateurs. Vous pouvez, si le cœur vous en dit, utiliser un autre port, auquel cas les clients devront le spécifier dans l'URL (*http://www.mondomaine.com:8100* pour un Port 8100). Là encore, laissez la valeur par défaut.

• *User www-data* et *Group www-data* précise l'identité complète (nom et groupe) de l'utilisateur Apache. La valeur par défaut en fonction de votre distribution sera *www-data.www-data* sur une Debian ou *wwwrun.nogroup* pour une distribution SuSE (par exemple). Ces valeurs par défaut ont été définies par le processus d'installation du package. La cohérence utilisateur existant/configuration d'Apache devrait parfaitement être respectée. Gardez cependant à l'esprit que les fichiers de données et les répertoires devront, en principe, appartenir à cet utilisateur. Bien évidemment, un fichier de données appartenant à un autre utilisateur et possédant les permissions de lecture adéquates conviendra, mais vous risquez de rencontrez des problèmes en raison de l'incohérence des informations. Il s'agit sans doute de votre premier serveur HTTP, accordez donc une grande importance à la "propreté" de la configuration.

• *ServerAdmin webmaster@mondomaine.com* permet de spécifier une adresse email pour le responsable du serveur HTTP (le webmaster ou webmestre en bon franglais). Il est fortement recommandé de définir une adresse spécifique dans la mesure du possible car c'est cette adresse qui recevra les mails concernant d'éventuels problèmes techniques du serveur. Evitez donc de noyer ces informations souvent capitales dans la masse des mails que vous recevez quotidiennement. Suivant le même principe, évitez de choisir ici root@mondomaine.com.

L'administrateur système (même si c'est vous) reçoit habituellement des messages concernant des problèmes plus importants à propos du fonctionnement du serveur. Sachez imposer un ordre et partir sur de bonnes bases.

• *ServerRoot /etc/apache* désigne le répertoire racine où sont placés, entre autres, les fichiers de configuration. Cela peut vous paraître étrange à première vue puisque le présent fichier est justement dans ce répertoire. Mais ce paramètre désigne également l'endroit par défaut pour d'autres types d'informations comme les journaux d'erreur et d'activité du système.

• *BindAddress* * spécifie l'adresse IP de l'interface à écouter pour la réception des requêtes HTTP. L'argument peut être une adresse, un nom de domaine ou, comme ici, le signe * spécifiant toutes les interfaces. En principe, si votre machine fait également office de passerelle/NAT pour l'accès à Internet, vous pourrez avec * aussi bien accéder au serveur depuis le réseau local (LAN) avec l'adresse IP de l'interface en question que via Internet avec l'adresse IP de l'autre interface

• *LoadModule module.so* vous permet d'utiliser les fonctionnalités modulaires d'Apache. En effet, le serveur HTTP peut être, lors de sa configuration/compilation, construit de manière statique (toutes les fonctionnalités choisies dans un seul binaire) ou de manière dynamique. Il est très peu probable que votre distribution utilise une version statique d'Apache et ce paramètre de configuration vous permettra donc d'ajouter ou de supprimer des fonctionnalités sans recompiler le serveur. Normalement, le fichier de configuration fourni et installé avec le package charge tous les modules nécessaires à un fonctionnement normal du serveur. Reportez-vous directement à la documentation de chaque module (`/usr/doc/apache`) pour connaître leur utilité. Vous n'avez en principe pas à toucher ces lignes (en ce qui concerne le support PHP, il s'agit également d'un module, reportez-vous à l'article correspondant dans le présent hors série).

• *PidFile /var/run/apache.pid* est le fichier où Apache va stocker son numéro de processus (PID). Ce paramètre est, en principe, déjà présent dans votre configuration exemple. Vous n'avez pas besoin d'y toucher. Il en va de même pour LockFile `/var/run/apache.lock`.

• *ServerName nommachine* définit un nom de serveur pour votre machine. Ce paramètre n'est à utiliser que si vous souhaitez définir un nom de machine à envoyer au client qui est différent du nom réel de la machine. En réalité, vous ne devrez utiliser ce paramètre que si le service Apache refuse de se lancer avec un message d'erreur ressemblant à ceci :

```
Starting web server: apache.
apache: cannot determine local host name.
Use the ServerName directive to set it manually.
/usr/sbin/apachectl start: httpd could not be
started
```

Ceci survient si le serveur n'arrive pas à déterminer son nom.

• *Timeout 300* détermine le temps limite pour les transactions HTTP (requêtes et réponses). Ceci permet d'éviter de rester indéfiniment en liaison avec des clients et ainsi consommer inutilement des ressources système.

• *KeepAlive On*, *MaxKeepAliveRequests 100* et *KeepAliveTimeout 15* permettent respectivement le support de connexion persistante, le nombre maximal de requêtes sur une même connexion et le temps limite pour une autre requête sur la même connexion. Il s'agit ici de paramètres permettant d'augmenter les performances de votre serveur HTTP. Les connexions HTTP utilisent un port TCP. TCP est un protocole avec gestion de connexion. Cela signifie qu'une communication TCP utilise un suivi de connexion. Sans entrer dans le détail, les données circulant sont fractionnées et numérotées. Si un fragment manque à l'appel, le destinataire demandera à l'expéditeur une nouvelle copie selon un principe prédéfini. Un client se connectant sur votre serveur peut donc profiter de ce principe pour que plusieurs requêtes fassent partie de la même connexion et ce en fonction des paramètres que vous aurez définis par ces arguments de configuration. Les valeurs par défaut permettent d'obtenir une performance plus importante mais ne sont peut-être pas optimales. Vous ne devrez toucher à ces options que si vous avez une connaissance suffisante du fonctionnement d'un réseau TCP/IP. Dans le cas contraire, satisfaisez-vous des paramètres par défaut.

• *StartServers 5*, *MinSpareServers 5* et

MaxSpareServers 10 sont également des paramètres permettant d'optimiser le fonctionnement de votre serveur HTTP. Apache, comme d'autres serveurs, lance plusieurs copies de lui-même au démarrage du service. Cela lui permet d'avoir un temps de réaction plus court lors de plusieurs connexions simultanées. Le paramètre *StartServers* définit le nombre de serveurs à lancer par défaut. Les deux paramètres suivants concernent respectivement le nombre minimal et maximal de serveurs à conserver en mémoire. Vous vous doutez que les valeurs par défaut sont utilisables sur n'importe quel type de matériel. Si vous montez un serveur destiné à recevoir un très grand nombre de visites simultanées, vous devrez augmenter ces trois valeurs et, bien sûr, prévoir une configuration matérielle (CPU, RAM) en conséquence.

• *MaxClients 150* et *MaxRequestsPerChild 30* représentent la continuité des paramètres précédents. Ils définissent respectivement le nombre maximum de clients connectés et le nombre maximal de clients par copie du processus serveur. Ainsi, tout en restant dans la limite des 150 clients, si une copie du processus dépasse les 30 clients, une nouvelle copie prendra le relais. Il est important de correctement définir les limites car en spécifiant des valeurs trop élevées ici et dans les paramètres précédents, vous pouvez en cas de surcharge de travail, faire effondrer tout le système. Mieux vaut alors refuser des connexions de la part de nouveaux clients plutôt que de conduire tout le système à sa perte.

Bien qu'il ne s'agisse que de paramètres de surveillance, nous avons choisi ici de placer les arguments de configuration qui vont suivre à part. Il s'agit de paramètres concernant les journaux d'activité du serveur Apache. Ceci peut paraître secondaire au premier abord mais ces journaux représentent la seule source à votre disposition pour la détection, l'analyse et la résolution d'éventuels problèmes. Ceci est particulièrement important lors de la mise en œuvre de votre premier serveur HTTP. Vous pourrez en apprendre beaucoup en parcourant ces journaux et ce, aussi bien sur la popularité de vos sites que sur des problèmes que vous n'auriez pas découverts d'une autre manière.

Les journaux peuvent, en effet, enregistrer toutes les transactions entre le serveur et les clients. Vous aurez ainsi une vision globale mais détaillée de l'activité et du rendement de vos sites. Les paramètres spécifiés dans `httpd.conf` sont des valeurs par défaut qui pourront être rendues obsolètes par d'autres fichiers configuration (voir `srm.conf` plus loin dans l'article). Ce n'est qu'en l'absence d'autres paramètres spécifiques aux sites (aux données) qu'ils seront pris en compte.

• *ErrorLog /var/log/apache/error.log* est le journal d'erreur. C'est sans aucun doute le fichier le plus important et celui, bien sûr, qui devra rester vide. En spécifiant un chemin non absolu (pas de / en début de ligne), la valeur du paramètre *ServerRoot* sera utilisée. De ce fait, en spécifiant simplement `error.log`, vous désignerez implicitement `/etc/apache/error.log`

• *LogLevel warn* définit le niveau d'importance des informations à placer dans le journal d'activité du serveur. L'argument utilisable sera choisi parmi les valeurs suivantes (du moins important au plus important) : `debug` (tout), `info`, `notice`, `warn` (avertissements), `error`, `crit` (information critique), `alert`, `emerg` (information urgente, capitale). La valeur `warn` inscrite par défaut est un bon choix pour signaler des avertissements.

• *LogFormat* est une directive permettant de définir des formats de lignes dans un journal personnalisé. Un certain nombre de formats prédéfinis lors de l'installation du package Apache sont sans doute placés dans votre fichier `httpd.conf`. La syntaxe de définition de ces formats est : *LogFormat "chaine de description" nom.*
Vous n'avez normalement pas besoin de définir vos propres formats de journaux, ceux par défaut offrant suffisamment de possibilités.

• *CustomLog /var/log/apache/nom.log nom_format* vous permettra de faire le lien entre un nom de fichier où stocker les informations du journal d'activité et le format à utiliser. La valeur par défaut pour ce paramètre sur une Debian est :

```
CustomLog /var/log/apache/access.log common
```

Un fichier `access.log` sera donc rempli en utilisant des lignes répondant au format `common`. Mais rien ne vous empêche d'ajouter un autre journal plus détaillé avec :

```
CustomLog /var/log/apache/access_total.log full
```

• *HostnameLookups on*, par défaut à *off*. Ce paramètre, une fois activé, vous permettra de voir appa-

raître dans les journaux non plus simplement l'adresse IP des clients se connectant, mais également leur nom d'hôte complet. Cela rend les journaux plus lisibles et vous permettra d'établir des statistiques plus précises concernant la provenance des connexions (par pays ou par fournisseur d'accès, par exemple).

2) **srm.conf**

A présent que notre serveur est configuré, nous pouvons nous intéresser aux données qui y seront placées. Les paramètres qui vont suivre concernent principalement les données mais certaines options peuvent écraser les valeurs du fichier précédent :

• *DocumentRoot /var/www* désigne tout simplement la racine de votre serveur HTTP. Un fichier `/var/www/toto.html` deviendra pour le client *http://www.mondomaine.com/toto.html*. Si vous avez prévu de placer `/var` sur un disque différent de la racine, assurez-vous d'avoir suffisamment d'espace à votre disposition. Cela vous évitera d'avoir à définir un autre emplacement racine et à copier les fichiers.

• *DirectoryIndex index.html* est le fichier de référence pour tout répertoire. Entendez par là que si l'utilisateur spécifie uniquement un nom de répertoire dans son URL, la sélection d'un fichier à retourner au client se fera en fonction de ce paramètre. Il est possible de spécifier plusieurs fichiers index en utilisant un espace entre les différents noms :

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

De cette manière, Apache retournera les informations contenues dans un fichier, s'il est trouvé dans le répertoire. L'ordre selon lequel vous placerez les noms en arguments de cette option détermine l'ordre de recherche. La recherche cesse dès qu'un fichier est trouvé.

D'autres paramètres méritent des explications plus approfondies :

a) Le *fancy indexing*

En temps normal, lorsqu'un client accède à un répertoire sans spécifier de nom de fichier, Apache cherche tout d'abord à fournir un fichier de remplacement grâce à *DirectoryIndex* puis, si cette recherche échoue, il envoie au client des données HTML listant le contenu du répertoire. Il s'agit d'une simple liste de noms de fichiers tels qu'ils apparaissent dans le répertoire en question (**figure 2**).

Apache met à votre disposition un mécanisme permettant d'embellir cette liste par le biais d'icônes spécifiant le type de fichier dont il s'agit, d'une date de modification, d'une taille et d'une description : le *fancy indexing* (**figure 3**).

Fig 2

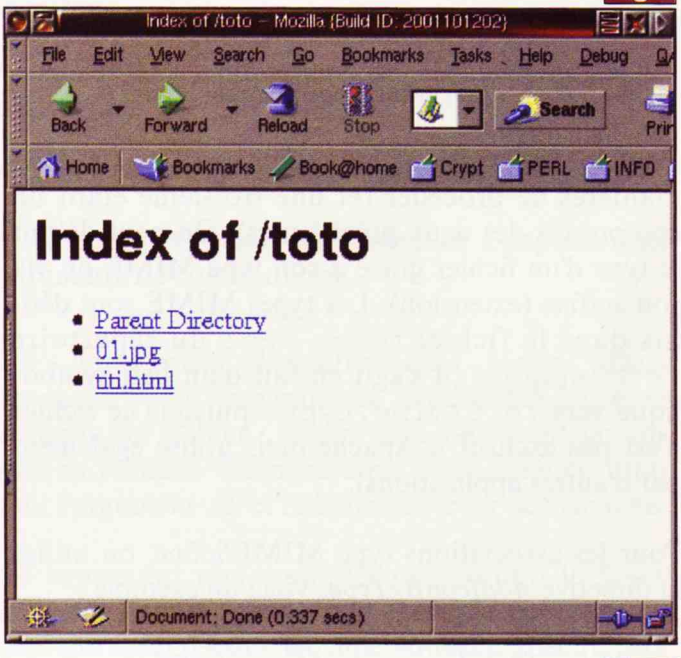
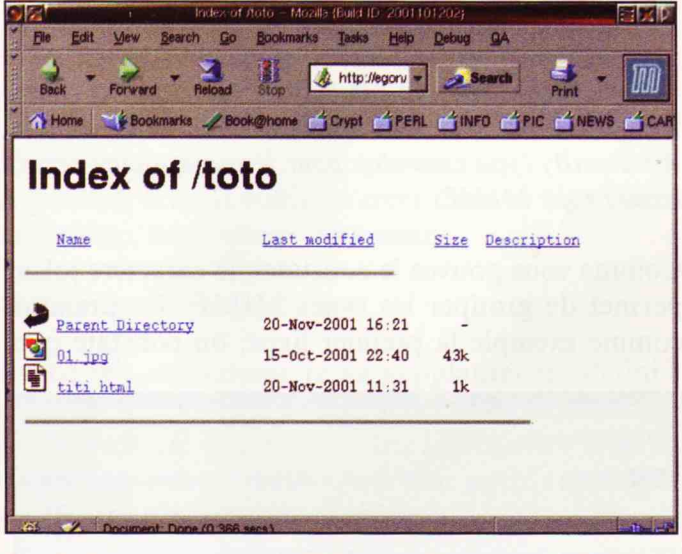


Fig 3



Les icônes permettant d'illustrer les types de fichiers sont placées normalement dans le répertoire `/usr/share/apache/icons`. Il s'agit d'une collection de quelques 80 fichiers au format GIF. Afin de faciliter les associations type/icône, la première étape consiste à créer un alias :

```
Alias /icons/ /usr/share/apache/icons/
```

Nous ferons dès lors référence au répertoire contenant les icônes en utilisant simplement `/icons`. Cela facilite non seulement la rédaction des lignes qui vont suivre, mais offre l'avantage de permettre l'utilisation de plusieurs jeux d'icônes. Imaginez un instant que vous dessiniez vous-même 80 autres images. Inutile alors de remplacer tous les noms de fichier dans `srm.conf`, il vous suffira de placer ces nouveaux fichiers dans `/usr/share/apache/mes_icones` et de changer l'alias en :

```
Alias /icons/ /usr/share/apache/mes_icones/
```

Pour les associations icône/fichier, il existe deux manières de procéder (et une troisième étant un compromis des deux précédentes). On peut définir le type d'un fichier grâce à son type MIME ou via son suffixe (extension). Les types MIME sont définis dans le fichier `mime.type` du répertoire `/etc/apache` (il s'agit en fait d'un lien symbolique vers `/etc/mime.types` puisque ce fichier n'est pas exclusif à Apache mais utilisé également par d'autres applications).

Pour les associations type MIME/icône, on utilise la directive *AddIconByType*. Voici un exemple :

```
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
```

La syntaxe est :
AddIconByType (mention pour les navigateurs texte, icône) type MIME

Comme vous pouvez le constater, le caractère joker permet de grouper les types MIME. En prenant comme exemple la premier ligne, on constate que

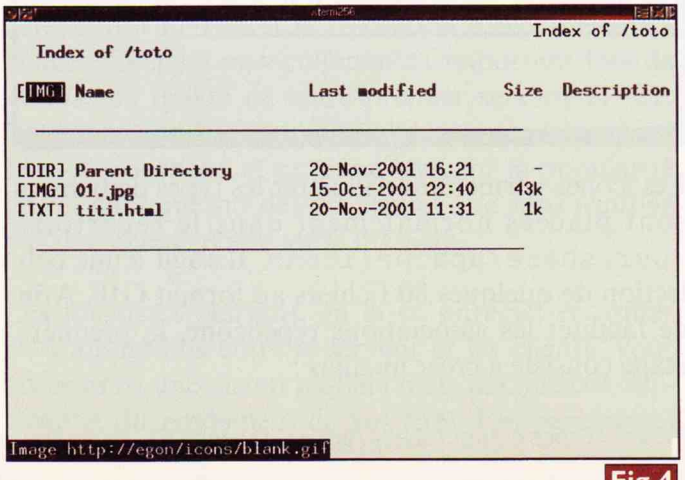


Fig 4

les types *text/plain, text/x-csrc, text/x-vCard*, etc. sont groupés et recevront une icône commune `text.gif`. Si un navigateur en mode texte (comme Links ou W3M) vient à consulter ce répertoire, la mention *TXT* sera utilisée à la place de l'image (**figure 4**). Il s'agit en fait d'un commentaire sur l'image (tag *ALT* en HTML).

La seconde possibilité d'association concerne l'utilisation du suffixe des fichiers comme référence. On utilise alors *AddIcon* :

```
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
```

Ici, la syntaxe est plus évidente puisqu'on se limite à préciser une icône suivie d'un ou plusieurs suffixe(s) s'y rapportant.

Enfin, la dernière solution consiste à définir un nouveau type dans `srm.conf` lui-même avec la directive *AddEncoding* :

```
AddEncoding x-gzip gz
```

Les formats de compression ne sont pas considérés comme des types MIME ; il faut en réalité parler d'encodage. C'est pourquoi vous ne trouverez normalement aucune mention de ce type de fichier dans votre `mime.types`. Une fois cette définition faite (ici pour les fichiers compressés avec l'utilitaire Gzip), il ne vous reste plus qu'à faire l'association avec une icône de la même manière que pour un type MIME :

```
AddIconByEncoding (CMP,/icons/compressed.gif) x-
gzip
```

Dernier point de configuration, comme personne ne peut prétendre être exhaustif dans ce genre de situation, il convient de définir des valeurs par défaut pour les types MIME :

```
DefaultType text/plain
```

et pour les icônes :

```
DefaultIcon /icons/unknown.gif
```

Enfin, nous n'oublierons pas d'activer le *fancy indexing* avec le paramètre adéquat :

```
FancyIndexing on
```

b) .htaccess

L'utilisation des fichiers de configuration httpd.conf et srm.conf nécessite le redémarrage du service Apache en cas de modifications. Afin d'éviter ce genre de désagrément, il est possible d'utiliser un fichier spécifique permettant de supplanter les directives de configuration. Ce fichier est connu généralement comme étant .htaccess mais ce nom n'est absolument pas obligatoire puisque le paramètre suivant permet de définir un autre nom :

```
AccessFileName .htaccess
```

Bien sûr, il est parfaitement inutile de sortir des sentiers battus en utilisant un .pouet ou un .titi en lieu et place du nom communément utilisé. Ce genre de fantaisie est absolument à éviter puisque si vous rencontrez un problème de configuration, les personnes susceptibles de vous aider vous parleront de .htaccess et ne manqueront pas de vous faire remarquer que votre comportement est des plus étrange.

Ce fichier .htaccess vous permet de spécifier des paramètres de configuration qui seront spécifiques sans redémarrer le serveur HTTP. Le paramètre *AccessFileName* concerne le site dans son ensemble. Vous ne pouvez utiliser *AccessFileName* que de manière globale. C'est ce nom qui sera utilisé pour les fichiers d'accès de tous les répertoires. Bien sûr, vous pouvez avoir un .htaccess spécifique à chaque répertoire, mais vous ne pouvez pas avoir un .htacces pour le répertoire toto et un .monaccess pour un autre répertoire titi. Il n'y a d'ailleurs aucun intérêt à cela...

Attention ! Pour que vous puissiez utiliser pleinement le mécanisme .htaccess, vous devez vérifier dans le fichier /etc/apache/access.conf la ligne contenant *AllowOverride*, en particulier sur une distribution Debian. Ce paramètre détermine l'étendue des possibilités offertes par .htaccess. Par défaut, la ligne est :

```
AllowOverride None
```

Ceci ne confère aucune autorité aux informations placées dans vos fichiers .htaccess. Les arguments de ce paramètre peuvent être :

- *None* pour aucune autorité ;
- *AuthConfig* pour l'écrasement des paramètres *AuthDBMGroupFile*, *AuthDBMUserFile*, *AuthGroupFile*, *AuthName*, *AuthType*, *AuthUserFile* et *require* ;
- *AuthUserFile* pour *AuthName*, *AuthType* et *require* ;
- *FileInfo* pour *AddType*, *AddEncoding* et *AddLanguage* ;
- *Indexes* pour tout ce qui concerne le *fancy indexing* ;
- *Limit* pour les limitations des IP où noms d'hôte des clients ;
- *Option* pour tout ce qui concerne les CGI ;
- *All* pour permettre l'écrasement de n'importe quel paramètre de configuration.

En laissant *AllowOverride None* dans access.conf, le paramètre *AccessFileName* et le contenu du fichier .htaccess ne seront jamais pris en compte. Changez donc cette ligne en utilisant l'argument *All* et redémarrez le service Apache.

En reprenant nos exemples concernant le *fancy indexing*, vous pouvez par exemple écraser une configuration d'icône par une nouvelle, spécifique à un répertoire. Ainsi, si nous créons un répertoire /var/www/toto sur notre serveur (et donc accessible par *http://www.mondomaine.com/toto*) et que nous y placions un fichier JPEG quelconque, celui-ci apparaîtra avec le *fancy indexing* accompagné d'une icônes image2.gif (par défaut).
A présent, il nous suffit de créer dans ce répertoire un fichier .htaccess contenant :

```
AddIcon /icons/tar.gif .jpg
```

De ce fait, ce paramètre va supplanter les définitions du srm.conf et **sans redémarrer** Apache. Il nous suffira d'accéder une nouvelle fois à *http://www.mondomaine.com/toto* pour constater que l'icône a effectivement changée.

Cette modification n'est donnée qu'à titre d'exemple ; en temps normal .htaccess est utilisé pour sécuriser l'accès à un répertoire. Nous verrons cela un peu plus loin.

c) *access.conf*

Ce fichier concerne directement les modalités d'ac-

cès aux répertoires de votre serveur HTTP. C'est ici que nous avons spécifié la portée des indications contenues dans les `.htaccess` des différents répertoires. Dans `access.conf`, les chemins mentionnés sont des chemins absolus (en partance de la racine du système de fichier).

Une directive très importante de ce fichier est :

```
<Files .htaccess>
order allow,deny
deny from all
</Files>
```

Ces simples lignes permettent d'interdire l'accès à n'importe quel fichier `.htaccess` quel que soit la provenance de la demande et le répertoire où est placé le fichier. Les deux directives importantes ici sont *order* et *deny/allow*. Voyons-les dans le sens inverse de leur apparition dans le fichier de configuration pour comprendre plus facilement leur intérêt :

• *allow* et *deny* permettent de respectivement autoriser ou refuser l'accès au contenu d'un fichier ou d'un répertoire. *allow* et *deny* sont suivis d'une condition. Ainsi, *deny from www.autredomaine.com* ou *deny from 213.11.3.129* interdira l'accès à la ressource désignée pour les clients provenant de ce nom de machine ou de cette adresse IP. Il est également possible d'utiliser un masque pour les adresses IP. Ainsi *deny from 213.11.37.0/255.255.255.0* interdira l'accès à la ressource pour tous les clients dont l'adresse débute par 213.11.37.

Le pendant à *deny from* est, bien sûr, *allow from,* qui autorisera l'accès à la ressource (fichier ou répertoire)

• *order* détermine l'ordre d'évaluation des conditions. Cela devrait répondre à une question que vous êtes sans doute en train de vous poser. Quel ordre prévaut entre *deny* et *allow* ? La réponse est : aucun. L'ordre d'apparition des directives *allow* ou *deny* n'a aucune importance. Il est alors nécessaire de spécifier cet ordre manuellement avec la directive *order*.

Pour bien comprendre le fonctionnement d'*order*, mettons en scène une configuration particulière : nous avons un réseau 192.168.0.0 sur lequel se trouve notre serveur en 192.168.0.1 et deux clients respectivement en 192.168.0.10 et 192.168.0.51. Nous allons définir une politique d'accès au fichier `titi.html` :

```
<Files titi.html>
order allow,deny
allow from 192.168.0.0/255.255.255.0
deny from 192.168.0.10
</Files>
```

Suivez bien, c'est important. La méthode la plus simple pour assimiler le fonctionnement est de tout simplement dire ce que nous voyons. *order* nous indique de vérifier la condition *allow* puis *deny*, ce qui nous donne : "*nous autorisons l'accès à tout le réseau 192.168.0.* à moins que nous interdisions l'accès à 192.168.0.10*" (ce n'est pas très français mais c'est clair). En vérifiant, il s'avère que 192.168.0.51 peut accéder à `titi.html` et 192.168.0.10 non, avec comme seule réponse un message très explicite "*You don't have permission to access /titi.html on this server*". C'est exactement ce que nous voulons.

A présent, voyons ce qui se passerait en inversant les arguments de la ligne *order* :

```
order deny,allow
```

Catastrophe ! 192.168.0.51 peut accéder au fichier mais 192.168.0.10 également ! Que s'est-il passé ? Encore une fois, il suffit de lire : "*Nous interdisons l'accès à 192.168.0.10 à moins que nous n'autorisions l'accès à tous les réseaux 192.168.0.**". 192.168.0.10 fait partie du réseau 192.168.0.* au même titre que 192.168.0.51 ; il a donc l'accès au fichier `titi.html`.

Notre exemple se base sur un réseau local et un fichier qui n'a pas l'air d'avoir une importance capitale. Il en va tout autrement pour un serveur HTTP connecté à Internet dont certains fichiers sont considérés comme sensibles. Le risque est encore plus important en utilisant *deny/allow from all* qui implique "tout le monde" comme l'une des parties en présence. Le maître mot est donc "lecture" ! En transformant les lignes de votre fichier de configuration en une phrase où vous utilisez "à moins que" (quitte à le faire à haute voix), vous vous assurez la bonne protection de vos fichiers.

La directive <Directory>

Nous venons de voir qu'il était possible de spécifier, par répertoire, une configuration spécifique par l'in-

termédiaire d'un fichier spécifique. Il existe cependant une autre manière de procéder, qui consiste directement à paramétrer les répertoires dans le fichier `access.conf`.

L'avantage de passer par les fichiers de configuration généraux concerne surtout les petites configurations. En effet, les fichiers `httpd.conf`, `access.conf` et `srm.conf` ne sont lus qu'au démarrage du service. Avec le système se basant sur `.htacces`, ce fichier spécifique au répertoire est recherché et parcouru à chaque fois qu'un client accède au répertoire. Cela constitue une surcharge de travail non négligeable pour un serveur basé, par exemple, sur un processeur i486 ou équivalent.
Ne sous-estimez pas le nombre de visites simultanées sur votre site et utilisez la configuration la plus sage pour votre matériel au moindre doute.

Les hôtes virtuels

L'association nom de domaine/adresse IP n'est pas exclusive. En d'autres termes, plusieurs sites Web peuvent être hébergés sur une seule et même machine possédant une seule adresse IP. Si nous prenons deux hôtes *www.premier.com* et *www.second.com*, il est parfaitement possible et normal que la résolution DNS nous donne la même adresse IP, 213.11.37.129 par exemple.

En l'absence de configuration spécifique, un client qui utilisera un de ces noms d'hôte comme URL arrivera sur le même fichier index qu'en utilisant l'autre nom d'hôte. Il n'y a donc pas grand intérêt à multiplier les noms de domaine ou d'hôte sans support spécifique de la part du serveur HTTP. Précisons que la différenciation porte bien sur le nom d'hôte complet et non seulement sur le domaine. Vous pouvez avoir ainsi un nom d'hôte *www.mondomaine.com* et un autre *perso.mondomaine.com*.

Heureusement pour nous, Apache est parfaitement capable de gérer de manière optimale plusieurs noms pour une seule machine (qui en aurait douté !). Ce système de différenciation est appelé l'utilisation d'hôtes virtuels puisque, physiquement, il n'y a qu'une seule machine, mais que les clients semblent accéder à des serveurs différents.

La mise en œuvre d'hôtes virtuels est très simple, il suffit d'utiliser les directives suivantes :

```
<VirtualHost www.mondomaine.com>
...
</VirtualHost>

<VirtualHost perso.mondomaine.com>
...
</VirtualHost>
```

Il ne vous reste plus, ensuite, qu'à compléter les configurations pour chaque hôte avec ses paramètres spécifiques dans `srm.conf` :

```
<VirtualHost www.mondomaine.com>
  DocumentRoot /var/www1
  DirectoryIndex index.php
  ErrorLog /var/log/apache/www-error_log
  CustomLog /var/log/apache/www-access.log common
</VirtualHost>
```

Nous configurons ici notre premier hôte virtuel et spécifions dans un premier temps la racine du site et directement ensuite le fichier index. Afin de faciliter la détection d'erreur et surtout de faire des statistiques distinctes pour chaque hôte virtuel, nous spécifions des journaux séparés pour chaque hôte. Notez que tous les paramètres utilisés dans les lignes précédentes nous sont déjà connus à ce stade de l'article.

Dernier point important ici concernant les modalités d'accès aux répertoires, deux cas de figure sont possibles : soit utiliser une directive *<Directory /var/www1>...</Directory>* dans le corps du *<VirtualHost>*, soit utiliser ces mêmes paramètres dans `access.conf` en spécifiant le répertoire. Personnellement, je serais tenté de centraliser les informations concernant les hôtes virtuels dans un seul et même paragraphe, mais l'autre solution est également applicable. C'est une affaire de goût et de cohérence.

Gérer les accès

Nous avons appris jusqu'à présent à limiter l'accès aux fichiers et aux répertoires en fonction de la provenance des requêtes. Cela est certes pratique mais insuffisant pour une utilisation plus poussée. Un même utilisateur peut parfaitement se connecter à votre ou vos site(s) depuis plusieurs endroits. De la même manière, un client se connectant en utilisant une connexion Internet non permanente (par modem analogique) obtiendra à chaque connexion une nouvelle adresse IP et sans doute un autre nom d'hôte.
Il est donc nécessaire de proposer aux clients une

autre méthode pour s'identifier et s'authentifier auprès de votre serveur. La méthode la plus simple dans ce cas est de tout simplement utiliser le principe de l'identifiant et du mot de passe.

Cette authentification par mot de passe peut, comme toute restriction, être faite soit dans `access.conf` et avec les directives *<Directory>* ou *<Files>*, soit via le fichier `.htaccess`. Les problèmes de performance du serveur sont identiques sinon plus importants sur les petites configurations étant donné la masse travail à accomplir.

Voici tout d'abord un exemple de contenu d'un fichier `.htaccess` que nous détaillerons par après :

```
AuthType basic
AuthName "Zone secure"
AuthUserFile /etc/apache/users
Require valid-user
```

Quatre directives ou options sont nécessaires :

• *AuthType* est le type d'authentification demandé. `basic` est le paramètre communément utilisé car il est compatible avec tous les navigateurs. Il définit, entre autre, le fait que l'identifiant de l'utilisateur et son mot de passe circulent en clair sur le réseau, ce qui peut poser des problèmes de sécurité. Un autre type d'authentification est *Digest* ; il définit un échange de *hash* MD5 entre le client et le serveur. Malheureusement, le support de ce type d'authentification est encore trop rare parmi les clients HTTP pour l'utiliser à grande échelle. Si vous accordez une grande importance à la confidentialité des données transitant sur le réseau, la meilleure solution est encore d'utiliser une authentification basique couplée à un serveur utilisant SSL (voir l'article sur *HTTP over SSL*).

• *AuthName* détermine le "mode" dans lequel s'applique cette authentification. Il s'agit en premier lieu d'afficher clairement un message dans la boîte de dialogue apparaissant sur l'écran de l'utilisateur distant.

• *AuthUserFile* définit le fichier contenant les informations sensibles (identifiant et mot de passe).

• *Require* définit les utilisateurs autorisés à accéder au répertoire en cause. L'argument *valid-user* spécifie ici que n'importe quel utilisateur qui satisfait l'authentification par mot de passe est accepté. Si vous souhaitez ajuster avec davantage de précisions quel utilisateur peut accéder à ce répertoire, il vous suffira de spécifier son nom ici (en cas de plusieurs utilisateurs autorisés, séparez les noms par des espaces).

Il nous faut ensuite créer le fichier de mots de passe. Pour cela, un petit utilitaire livré dans le même package que le serveur Apache est mis à votre disposition. Il s'agit de *htpasswd*. Pour respecter l'exemple donné pour `.htaccess`, nous allons créer un fichier `/etc/apache/users` :

```
# htpasswd -c users moimeme
New password:
Re-type new password:
Adding password for user moimeme
```

Nous pourrons ensuite lister le contenu du fichier `users` nouvellement créé :

```
# cat users
moimeme:QUbeW02EGIwGU
```

Il nous est ensuite possible de tout aussi facilement ajouter de nouvelles entrées dans ce fichier avec, par exemple :

```
# htpasswd users machintruc
New password:
Re-type new password:
Adding password for user machintruc
```

Vous l'aurez compris, l'option `-c` ne sert qu'à créer un nouveau fichier. Dès lors où vous aurez procédé à ces modifications, toute tentative d'accès au répertoire où est placé le fichier `.htaccess` provoquera, chez le client, l'affichage d'une boîte de dialogue demandant un nom d'utilisateur (**figure 5**) et un mot de passe correspondant. Pour l'heure,

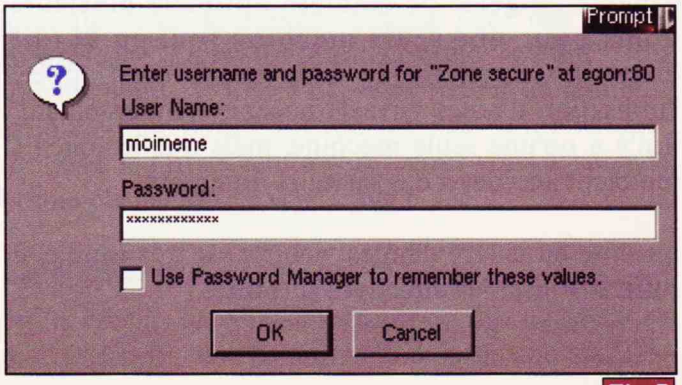


Fig 5

`moimeme` et `machintruc`, en spécifiant les informations nécessaires, peuvent tous deux accéder au contenu du répertoire. Si maintenant nous changeons la ligne

```
Require valid-user
```

en

```
Require moimeme
```

seul l'utilisateur `moimeme` pourra accéder au contenu en fournissant le mot de passe adéquat. `machintruc` avec ou sans mot de passe correct se verra refuser l'accès. Terminons cette partie en signalant qu'il est possible de conjuguer les directives d'authentification que nous venons de voir avec *deny* et *allow*. Il suffit pour cela de préciser la politique à adopter. Soit nous demandons à ce que les deux conditions soient remplies :

```
Satisfy all
```

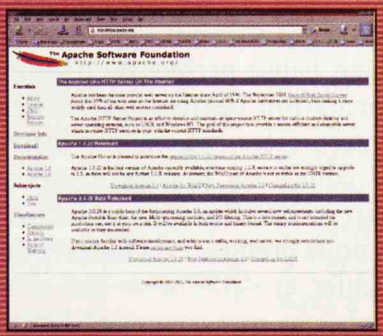
Soit une seule des conditions sera nécessaire pour autoriser l'accès :

```
Satisfy any
```

Nous arrivons au terme de notre apprentissage de base du serveur Apache. Avec les indications qui ont été données, vous devriez être en mesure de démarrer votre premier serveur et vos premières pages. N'oubliez pas de réfléchir posément avant de faire des choix techniques ou stratégiques. Mieux veut être prudent et progresser doucement mais sûrement..

Liens

Apache Home Page
http://httpd.apache.org/



HTML : HyperText Mark-up Language

Cet article s'adresse aux personnes n'ayant jamais touché à la composition de pages Web. Précisons de suite que les informations indiquées ici se rapportent exclusivement à la création de pages Web "à la main". Bien qu'il existe un grand nombre de logiciels permettant de réaliser facilement de telles pages, il est important de connaître la syntaxe et le fonctionnement du langage de description de pages Web : le HTML

L'acronyme, peu parlant pour des personnes non anglophones, nous indique que le HTML est un langage permettant de construire des documents hypertextes. Un tel document est plus qu'une mise en forme des informations. Un langage comme HTML permet d'établir des relations entre les pages de manière à permettre une navigation de l'une à l'autre. Tous les sites que vous visitez en tant qu'internaute sont écrits en HTML. Il s'agit d'un standard international qui permet à tous les navigateurs existants d'accéder à tous les sites de la planète.

Vous le verrez sans doute au fur et à mesure de vos expériences sur le terrain, HTML est un langage tolérant. Entendez par là que sa syntaxe souple permet aux navigateurs de tolérer les erreurs. Il n'en reste pas moins que le HTML répond à une norme stricte établie par un consortium du nom de W3C.

Les fichiers de description HTML sont de simples fichiers texte. Ces fichiers circulent sur le réseau Internet à la demande des navigateurs. Le protocole utilisé pour ces demandes et les réponses est HTTP. Voici un exemple stylisé de transaction entre un navigateur et un serveur HTTP :

navigateur : je voudrais le fichier *index.html*
serveur : j'envoie le fichier
navigateur : je lis le fichier, je l'interprète, puis je demande les éléments qui me manquent (images)
serveur : j'envoie les images
navigateur : la transaction est finie, j'analyse le fichier et affiche les informations à l'écran.

On constate ici très nettement que ce que vous voyez en surfant sur le Web n'est que l'interprétation des informations par votre navigateur. Voilà pourquoi il est nécessaire de respecter strictement les standards en usage si vous désirez que vos pages s'affichent devant l'internaute de la même manière avec tous les navigateurs du marché. Bien sûr, la norme étant elle-même très restrictive, vous devrez faire des concessions entre compatibilité et aspect final. Le langage HTML permet de composer des pages comprenant du texte, des liens, des listes, des tableaux, des images ou encore des caractères spéciaux. Nous allons voir cela en détail.

HTML de base

Un fichier HTML est un fichier texte ; vous pourrez donc le créer à l'aide de n'importe quel éditeur de texte sous Linux (Vim, Emacs, Nedit, CoolEdit, etc.). La structure de base d'un fichier HTML est la suivante :

```
<HTML>
<HEAD>
<TITLE>Mon titre de page</TITLE>
</HEAD>
<BODY>
Un texte sur le corps de la page.
</BODY>
</HTML>
```

Plusieurs informations très importantes apparaissent ici :

- Les mentions entre <> sont appelées balises (*tags* en anglais). Une balise permet de définir un paragraphe d'information destiné au navigateur. Les balises sont interprétées comme des instructions et n'apparaissent jamais sur la page. L'espace entre la balise de départ et la balise de fin définit la portée des balises. Si nous prenons <HTML>, qui marque le début du fichier, et </HTML> qui en marque la fin, nous définissons que toutes les lignes se trou-

vant entre ces deux balises sont à considérer comme du code HTML. Ce raisonnement s'applique quasiment à toutes les balises HTML (il y a de rares exceptions).

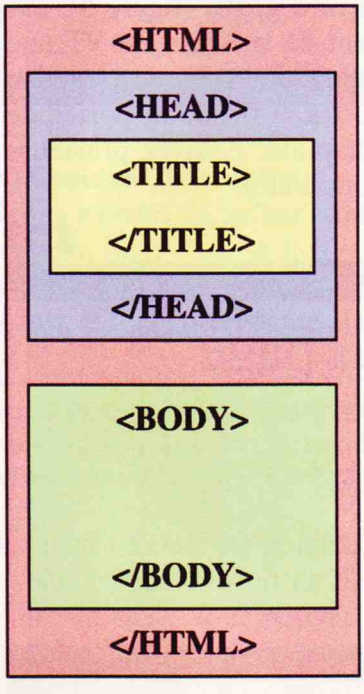
• La casse utilisée pour les balises n'a aucune importance. Vous pouvez parfaitement débuter votre code HTML avec <hTmL> et le terminer avec </htML>. Le navigateur ne signalera aucun problème. Il en va tout autrement en ce qui concerne la lisibilité de votre code. Garder une constance (toujours en majuscules ou toujours en minuscules) dans la rédaction de vos fichier les rendra plus lisibles et donc plus faciles à modifier.

• Il est possible d'imbriquer les portées des balises. On le voit dans le code que nous venons de donner, <HEAD></HEAD> se trouvent effectivement à l'intérieur de la portée de <HTML>. Et à l'intérieur de la porté de <HEAD> se trouvent les balises <TITLE></TITLE>.

• Les retours chariot ne sont pas importants en ce qui concerne les balises HTML. Le précédent code HTML aurait tout aussi bien fonctionné écrit de cette manière :
<HTML><HEAD><TITLE>Mon titre de page</TITLE></HEAD><BODY>Un texte sur le corps de la page.</BODY></HTML>

Un résumé de ces indications est fourni sur le schéma en **figure 1**.

Fig.1



Petite précision avant de poursuivre plus loin nos explications : le HTML est un standard développé par le W3C. Qui dit standard dit obligatoirement norme à respecter. Tout comme pour d'autres standards, il existe plusieurs versions des recommandations HTML. La version la plus courante est 3.2 et la version 4.01 tend à se généraliser. Nous ne vous proposerons pas ici de fonctionnalités spécifiques au HTML 4 et les indications valables pour le 3.2 le sont également pour le 4.01. J'ai utilisé le mot "recommandations" car il faut prendre en compte un paramètre de tolérance.

Les navigateurs Web ne respectent pas tous les recommandations du W3C. De la même manière, plus de la majorité des pages HTML disponibles sur Internet font abstraction de certaines mentions considérées comme obligatoires pour respecter le standard. Les navigateurs sont heureusement tolérants et corrigent eux-mêmes vos erreurs dans la mesure du possible. Ainsi, notre premier code exemple ne respecte pas le standard défini par le W3C (il n'en est pas moins parfaitement fonctionnel). Pour pallier ce problème, nous aurions dû ajouter une balise définissant le type et la version du document HTML et l'encodage utilisé pour les caractères. Un code HTML parfaitement conforme aux standards serait le suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
Final//EN">
<HTML>
<HEAD>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<TITLE>Mon titre de page</TITLE>
</HEAD>
<BODY>
Un texte sur le corps de la page.
</BODY>
</HTML>
```

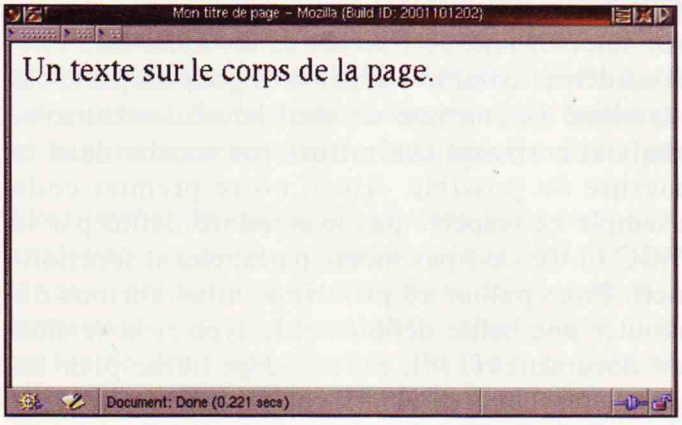
Dernier point sur les recommandations du W3C, le consortium met à votre disposition un moteur de vérification HTML : le *HTML Validator*. Il suffira de lui indiquer une URL pour que celui-ci vous signale le moindre problème ou, avec beaucoup de travail de votre part, de patience et de chance, un magnifique message "*No errors found! Congratulations, this document validates as HTML 3.2 !*". Le HTML Validator est accessible depuis **http://validator.w3.org/**. N'hésitez pas à l'utiliser sur vos pages mais également sur d'autres afin de bien comprendre les implications du respect ou du non-respect des standards.

Le texte dans HTML

En se basant sur notre exemple précédent, nous obtenons ce qui est sans doute le plus simple des

fichiers HTML (**figure 2**). Nous avons un titre affiché dans le nom de la fenêtre du navigateur (ici Mozilla) et une simple phrase dans le corps (les différentes barres d'outil et de navigation du navigateur ont été délibérément masquées pour faciliter les démonstrations).
Nous pouvons d'ores et déjà ajouter des éléments de formatage à ce texte.

Fig.2



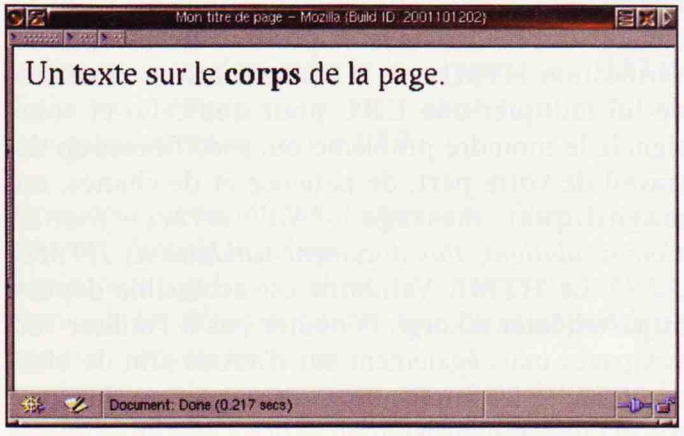
1) Mise en gras

Nous pouvons mettre en relief tout ou partie du texte en mettant certains éléments en caractères gras. Nous disposons de plusieurs balises pour cela :

```
Un texte sur le <b>corps</b> de la page.
```

Mais corps aurait permis d'obtenir le même rendu (**figure 3**). Tout comme pour les autres balises que nous avons déjà vu, la portée de la mise en gras est définie par la balise de départ () et celle de fin ().

Fig.3



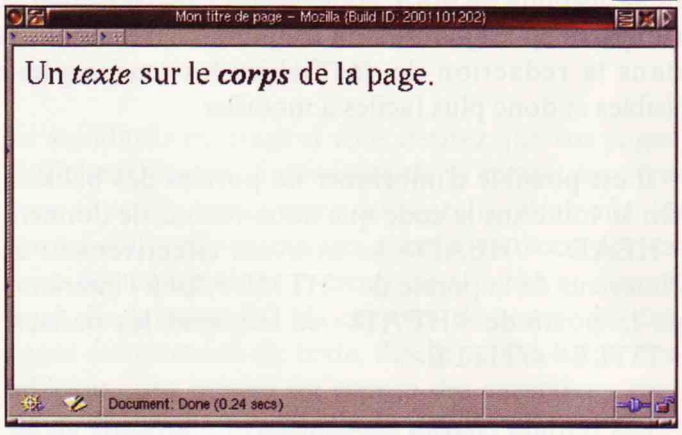
2) Italique

Pas grand-chose à dire de ce côté, ce qui est vrai pour le gras l'est également pour la mise en l'italique :

```
Un <i>texte</i> sur le <b><i>corps</i></b> de la page.
```

Vous le constaterez sur la **figure 4**, il est possible de cumuler l'italique et le gras. Essayez cependant de bien respecter les portées en jeu. corps vous donnera sans doute le même résultat, mais en procédant de la sorte vous vous dirigez tout droit vers un problème à moyen terme. Mieux vaut avoir de l'ordre...

Fig.4



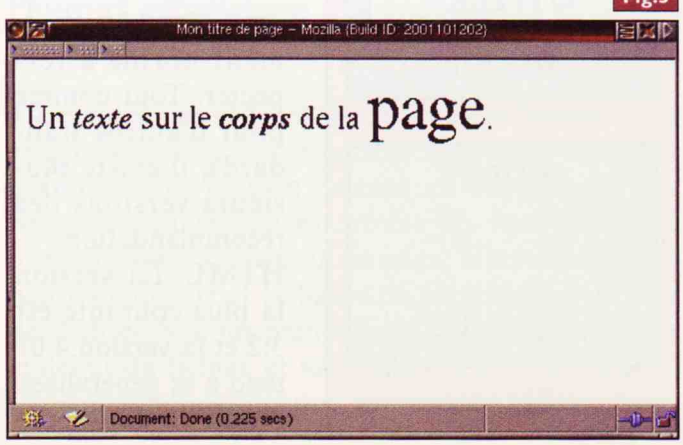
3) La balise FONT

Cette balise est un peu particulière étant donné qu'elle ne se suffit pas à elle-même. En effet, elle prend une option en argument. Essayez le code suivant :

```
Un <i>texte</i> sur le <b><i>corps</i></b> de la
<font size=6>page</font>.
```

Le mot "page" s'en trouvera grossi (**figure 5**), car nous passons en argument de la balise FONT une option permettant de définir une taille de caractère (`size=6`). Nous marquons la fin de la portée de FONT par la balise adéquate. Essayez plusieurs tailles, ça devrait venir tout seul :)

Fig.5

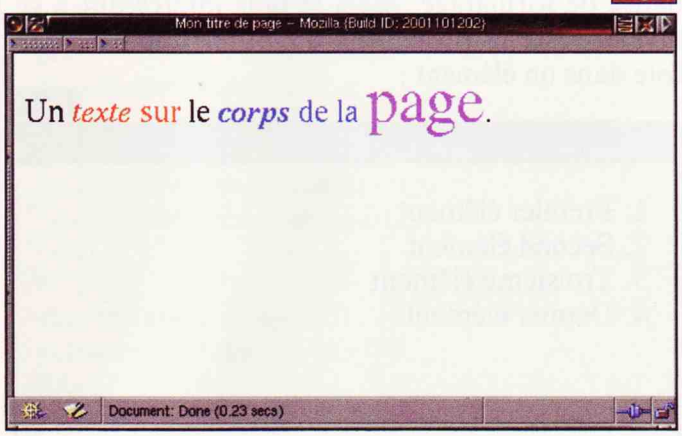


Mais `size` n'est pas le seul argument de la balise, vous pouvez de la même manière ajouter de la couleur à votre texte :

```
Un <font color="#FF0000"><i>texte</i> sur
</font>le <font
color="#0000FF"><b><i>corps</i></b> de la</font>
<font size=6 color="#BB55BB">page</font>.
```

Vous obtiendrez un rendu correspondant à la **figure 6**. La couleur est définie par l'option `color=` suivie d'un argument entre guillemets et précédée d'un dièse. La syntaxe utilisée pour définir une couleur est #RRVVBB où RR est une valeur de rouge entre 00 et FF, VV une valeur de vert et BB une valeur de bleu. La notation numérique utilisée est hexadécimale (base 16). Contrairement à la notation décimale entre 0 et 9, l'hexadécimale est comprise entre 0 et F, ce qui, sur deux positions, nous permet d'utiliser en décimales 256 valeurs entre 0 et 255 (00 à FF). #FFFFFF nous donnera du blanc et #000000 du noir. Je vous conseille d'essayer plusieurs combinaisons possibles pour bien saisir le fonctionnement de cette syntaxe.

Fig.6



4) Commentaires et sauts

Nous n'avons ici qu'une seule ligne. En HTML, un saut de ligne dans le fichier de description ne sera pas répercuté sur le rendu final dans le navigateur. Ainsi, la ligne précédente aurait pu être :

```
Un <font color="#FF0000"><i>texte</i>
sur</font> le <font color="#0000FF">
<b><i>corps</i></b> de la</font>
<font size=6 color="#BB55BB">page</font>.
```

Le rendu aurait été le même. Il faut donc spécifier explicitement un saut de ligne à l'aide de la balise
 (*break*). Vous pouvez cumuler les <br> pour obtenir des lignes blanches.

Une autre balise vous permet de faire un saut de ligne. Cette balise est habituellement mal utilisée puisque les navigateurs tolèrent une syntaxe incorrecte. Ce saut de ligne est en fait un saut de paragraphe. Un paragraphe est un texte placé entre les balises et . Il est très courant, cependant, que seule soit utilisé pour séparer deux paragraphes :

Syntaxe correcte :

```
<p>Voici un beau paragraphe.</p>
<p>Et en voici un autre.</p>
```

Syntaxe tolérée :

```
Voici un beau paragraphe.<p>
Et en voici un autre.
```

Lorsque votre fichier HTML comprend plusieurs dizaines ou centaines de lignes, il peut être utile de rajouter des commentaires, c'est-à-dire des mentions qui figureront dans le fichier mais qui n'auront aucune incidence sur le rendu de la page. Un commentaire est une chaîne de caractères placée entre <!-- et -->. La chaîne, le préfixe et le suffixe forment une balise HTML unique.
Exemple :

```
Un premier texte.<br>
<!-- un commentaire -->
Un second texte.
```

5) Balises HTML 3.2

Un certain nombre de balises sont définies par le standard HTML en version 3.2. Cependant, tous les navigateurs ne les supportent pas toutes. Voici quelques-unes d'entre elles :

```
<code>
Un code;
avec un langage;
quelconque();
</code>
```

Cette balise permet de changer de police et d'en utiliser une non proportionnelle (où tous les caractères possèdent la même largeur, y compris les espaces). Cette balise est habituellement utilisée pour faire mention d'un code ou d'une instruction dans le corps d'une phrase ou d'un paragraphe.

```
<pre>
Un code;
```

```
avec un langage;
quelconque();
</pre>
```

Cette balise indique le début et la fin d'un texte préformaté. Entendez par là que le navigateur considérera que les lignes doivent apparaître de la même manière qu'elles sont placées dans le fichier HTML. De ce fait, les espaces, les tabulations et les retours chariot sont comptés. Cette balise est également utilisée pour afficher du code mais cette fois, la mise en forme originale est conservée.

```
Voici une phrase avec un <u>mot</u> particulier.
```

La balise <u> permet de souligner un mot. Ceci est normalement à éviter sur vos futures pages car les mots ou les phrases étant également des liens vers d'autres pages sont habituellement soulignés d'office par les navigateurs.

```
Oh ! Cela <blink>clignote</blink>.
```

Comme le dit si bien l'exemple, <blink> permet de faire clignoter une portion de texte. On relèvera cependant que cela ne fonctionne qu'avec Netscape Navigator. Explorer et Mozilla ne prenant pas en compte la balise.

6) caractères spéciaux

En principe, seuls un certain nombre de caractères sont autorisés dans du code HTML. Ainsi, les caractères accentués chers aux francophones se voient recalés au rang de caractères spéciaux. Pour pouvoir faire apparaître en toute sécurité ces caractères, il faut utiliser une syntaxe particulière.

Le caractère "é", par exemple, deviendra *é* dans votre code HTML. Il en va de même avec "è" qui devient *è* ou encore "ç" qui devient *¸*. Encore une fois, il s'agit de recommandations. En utilisant cette syntaxe, vous vous assurez que vos caractères spéciaux seront correctement affichés avec n'importe quel navigateur, y compris ceux n'utilisant pas l'encodage Latin-1. Libre à vous d'estimer que les clients consultant vos pages disposeront de navigateurs capables d'afficher les caractères accentués placés tels quels dans votre code HTML. En ce qui concerne le présent article, nous avons choisi d'utiliser directement les caractères accentués dans les codes donnés en exemple. Ceci dans le seul but de faciliter la lecture desdits exemples et non de vous pousser à faire de même dans vos fichiers.

Les listes

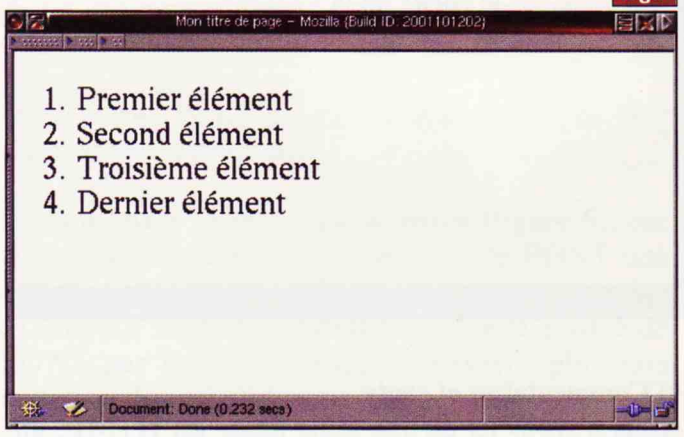
Lister une suite d'arguments, de mots, d'objets, etc. est une chose relativement facile en HTML. Nous devons cette facilité à l'absence de numérotation statique. La seule chose importante en créant une liste avec HTML est de bien structurer les éléments dans votre esprit avant de saisir les balises.

La portée d'une liste est définie par les balises <OL></OL> (OL = *Ordered List*). A l'intérieur de cette portée, nous devons spécifier le début et la fin de chaque élément de la liste. Voici un exemple :

```
<OL>
<LI>Premier élément</LI>
<LI>Second élément</LI>
<LI>Troisième élément</LI>
<LI>Dernier élément</LI>
</OL>
```

Le résultat est donné en **figure 7**. La liste est automatiquement numérotée sans aucune précision de votre part. Le texte dans la portée des balises <LI></LI> peut contenir n'importe quelle autre balise de formatage. Mais le plus intéressant à ce niveau reste la possibilité d'ajouter une nouvelle liste dans un élément :

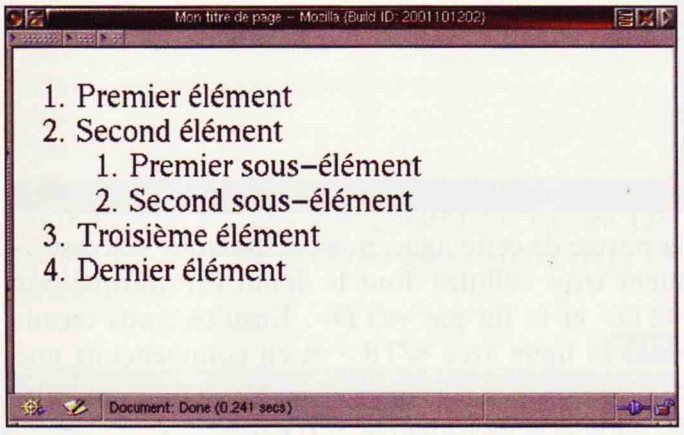
Fig.7




```
<OL>
 <LI>Premier élément</LI>
 <LI>Second élément
  <OL>
  <LI>Premier sous-élément</LI>
  <LI>Second sous-élément</LI>
  </OL>
 </LI>
 <LI>Troisième élément</LI>
 <LI>Dernier élément</LI>
</OL>
```

Nous avons ici ajouté dans le second élément une nouvelle liste ordonnée. Le résultat est automatiquement mis en page par le navigateur (**figure 8**). Vous conviendrez avec moi que la double numérotation peut être déroutante pour le client.

Fig.8

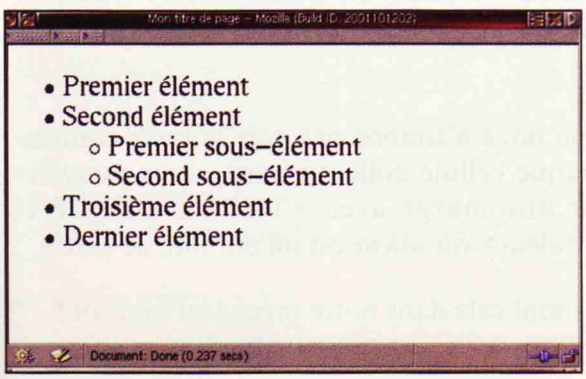


HTML met à notre disposition une autre paire de balises permettant de définir une liste non ordonnée, c'est-à-dire qui ne comporte pas de marque d'énumération numérotée. Ces balises <UL><UL> permettent de créer des listes utilisant des puces. Voici le précédant exemple utilisant <UL> au lieu de <OL> :

```
<UL>
 <LI>Premier élément</LI>
 <LI>Second élément
  <UL>
  <LI>Premier sous-élément</LI>
  <LI>Second sous-élément</LI>
  </UL>
 </LI>
 <LI>Troisième élément</LI>
 <LI>Dernier élément</LI>
</UL>
```

On constate immédiatement le changement (**figure 9**). Ces nouvelles balises présentent un avantage certain, l'aspect des puces change en fonction de l'imbrication des listes. Voici un ultime exemple d'imbrication pour bien asseoir ses connaissances :

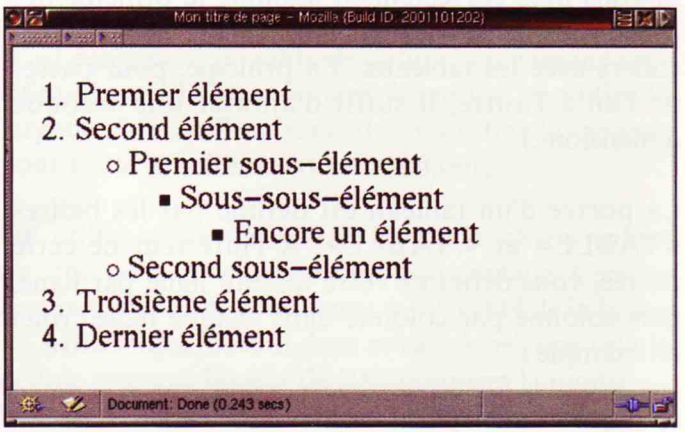
Fig.9




```
<OL>
 <LI>Premier élément</LI>
 <LI>Second élément
  <UL>
  <LI>Premier sous-élément
   <UL>
   <LI>Sous-sous-élément
    <UL>
    <LI>Encore un élément</LI>
    </UL>
   </LI>
   </UL>
  </LI>
  <LI>Second sous-élément</LI>
  </UL>
 </LI>
 <LI>Troisième élément</LI>
 <LI>Dernier élément</LI>
</OL>
```

Le résultat en **figure 10** se passe de commentaire, si ce n'est en remarquant que l'imbrication des listes montre des limites dans la variété des puces.

Fig.10



Les tableaux

Si vous pensez ne jamais utiliser de tableaux dans vos pages car vous n'avez pas d'information à présenter de cette manière, vous faite une grosse erreur. Les tableaux HTML ne permettent pas seulement de... faire des tableaux. Cela peut vous paraître absurde au premier regard mais quelques exemples arriveront sans mal à vous convaincre :

- Une mise en page HTML utilisant deux colonnes ou plus nécessite obligatoirement l'utilisation d'un tableau.

- La mise en avant d'un paragraphe à l'aide d'une couleur de fond spécifique est également le résultat de l'utilisation d'un tableau.

• Le placement d'images dans une page structurée se fait à l'aide d'un tableau.

• Bon nombre de sites mettent à l'écart une autre fonctionnalité HTML appelée *frame* au profit de l'utilisation de tableaux. Les *frames* permettent de diviser l'écran du navigateur de manière à obtenir plusieurs zones distinctes. Ajoutons à cela que très peu de navigateurs en mode texte savent gérer des *frames* et que ceux qui le font les transforment en tableau avant de les afficher.

• La création de belles pages est bien plus facile à l'aide de tableaux.

Les tableaux HTML sont beaucoup plus qu'un simple élément permettant de représenter des valeurs ou des caractéristiques dans une structure rectangulaire. Les tableaux HTML vous permettront de placer avec une grande précision les éléments de votre page.

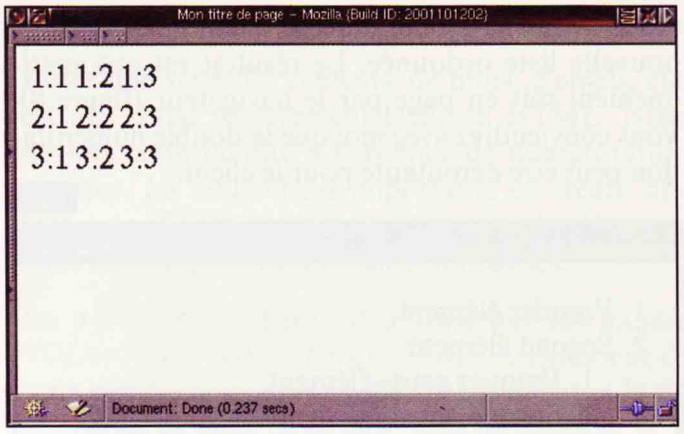
Si vous avez correctement assimilé le principe des listes, vous ne devriez pas avoir de problèmes particuliers avec les tableaux. En principe, pour passer de l'un à l'autre, il suffit d'ajouter une seconde dimension :)

La portée d'un tableau est définie par les balises <TABLE> et </TABLE>. A l'intérieur de cette portée, vous définirez votre tableau ligne par ligne, puis colonne par colonne dans chaque ligne. Voici un exemple :

```
<TABLE>
<TR>
 <TD>1:1</TD><TD>1:2</TD><TD>1:3</TD>
</TR>
<TR>
 <TD>2:1</TD><TD>2:2</TD><TD>2:3</TD>
</TR>
<TR>
 <TD>3:1</TD><TD>3:2</TD><TD>3:3</TD>
</TR>
</TABLE>
```

Le résultat est syntaxiquement correct (**figure 11**) mais absolument inacceptable en terme de rendu. Avant de nous pencher sur l'aspect esthétique de la chose, résumons ce que nous venons de faire. Nous avons commencé notre tableau avec <TABLE> puis directement la première ligne ou rangée. Dans

Fig.11



la portée de cette ligne, nous définissons successivement trois cellules dont le début est marqué par <TD> et la fin par </TD>. Ensuite, nous terminons la ligne avec </TR> et en commençons une nouvelle. Enfin, la dernière ligne composée, nous terminons le tableau avec </TABLE>.

Pourquoi notre tableau est-il aussi... "moche" ? En fait, chaque balise de départ peut accepter un certain nombre de paramètres optionnels :

1) <TABLE> se rapporte au tableau dans son ensemble.

• Nous pouvons préciser une largeur totale pour le tableau avec <TABLE WIDTH=valeur> où *valeur* est un nombre de pixels ou un pourcentage de la largeur totale de la fenêtre du navigateur. Ainsi, si nous souhaitons que notre tableau occupe l'ensemble de la largeur de la fenêtre, nous ferons <TABLE WIDTH="100%">.

• Nous pouvons demander l'affichage d'une bordure en utilisant <TABLE BORDER=valeur> où *valeur* est un nombre de pixels entre 0 (pas de bordure) et un nombre quelconque.

• Nous pouvons spécifier un espace en pixels entre la bordure de chaque cellule et la bordure du tableau avec <TABLE CELLSPACING=valeur>. Avec une valeur 0, vous obtiendrez un tableau classique.

• Comme nous n'aimons pas voir le texte contenu dans chaque cellule collé au bord, nous pouvons préciser une marge avec <TABLE CELLPADDING=valeur> où *valeur* est un nombre de pixels.

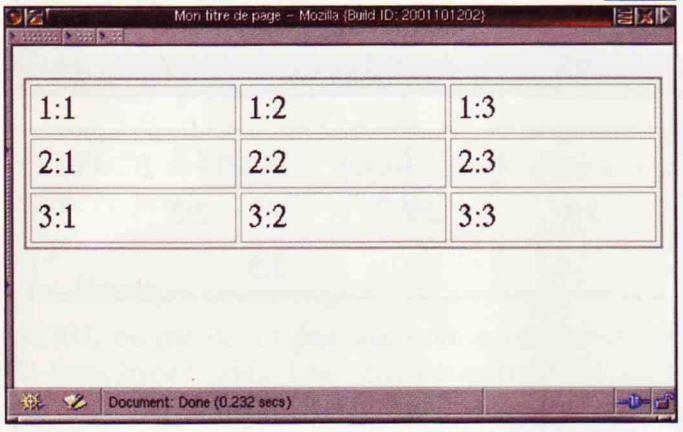
Incluons tout cela dans notre précédent tableau :

```
<TABLE WIDTH="100%" BORDER="2" CELLSPACING="3"
```

```
CELLPADDING="5">
<TR>
 <TD>1:1</TD><TD>1:2</TD><TD>1:3</TD>
</TR>
<TR>
 <TD>2:1</TD><TD>2:2</TD><TD>2:3</TD>
</TR>
<TR>
 <TD>3:1</TD><TD>3:2</TD><TD>3:3</TD>
</TR>
</TABLE>
```

Notre tableau est déjà plus présentable (**figure 12**). Mais il est possible d'affiner encore davantage sa présentation.

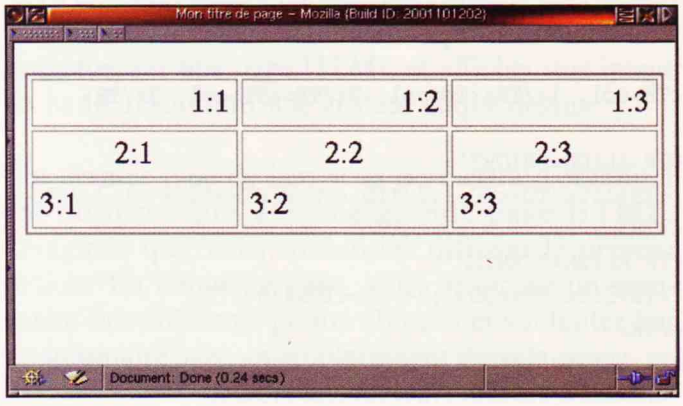
Fig.12



2) <TR> prend également un certain nombre d'options en compte. Celles-ci se rapportent aux lignes du tableau dans leur intégralité. Ainsi, pour un tableau de chiffres, par convention, il convient d'aligner le contenu des cellules à droite. Pour les textes, on choisira un alignement à gauche ou centré en fonction de la nature du contenu. L'alignement des cellules dans une ligne du tableau est défini par <TR ALIGN=mot clef> où *mot clef* peut être RIGHT (droite), LEFT (gauche) ou CENTER (centre). Notre tableau comporte trois lignes, nous pouvons donc avoir un aperçu de chaque alignement (**figure 13**) :

```
<TABLE WIDTH="100%" BORDER="2" CELLSPACING="3"
CELLPADDING="5">
<TR ALIGN="RIGHT">
 <TD>1:1</TD><TD>1:2</TD><TD>1:3</TD>
</TR>
<TR ALIGN="CENTER">
 <TD>2:1</TD><TD>2:2</TD><TD>2:3</TD>
</TR>
<TR ALIGN="LEFT">
 <TD>3:1</TD><TD>3:2</TD><TD>3:3</TD>
</TR>
</TABLE>
```

Fig.13

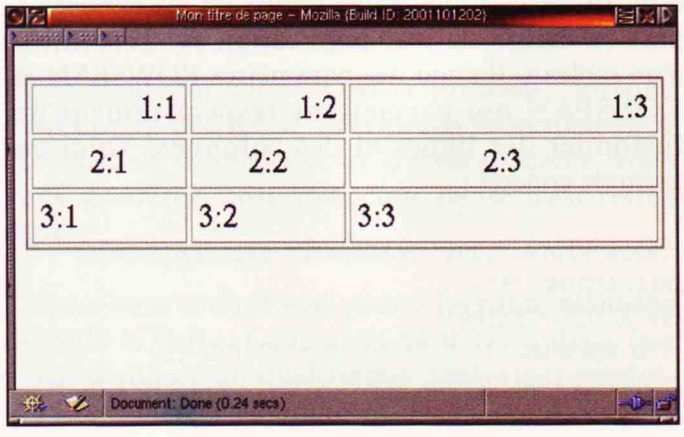


Un autre paramètre concerne l'alignement vertical. Les cellules d'une ligne peuvent ainsi être alignées en haut avec <TR VALIGN="TOP"> ou en bas avec <TR VALIGN="BOTTOM">.

3) <TD> prend en option des paramètres concernant chaque cellule. Les paramètres d'alignement horizontaux et verticaux sont les mêmes que pour <TR> mais ne s'appliquent qu'à la cellule courante. Il est également possible de définir une largeur spécifique pour chaque colonne. Attention, modifier la largeur d'une cellule revient à modifier la largeur pour toute la colonne correspondante.

On utilise alors <TD WIDTH=valeur> où *valeur* est une taille en pixels ou un pourcentage. Ceci ne provoquera pas de conflit avec le WIDTH de la balise <TABLE> qui prévaut dans ce cas. Ainsi, une cellule définie avec une largeur de 50% occupera la moitié du tableau et les autres colonnes se répartiront le reste (**figure 14**, la dernière colonne est à 50%).

Fig.14



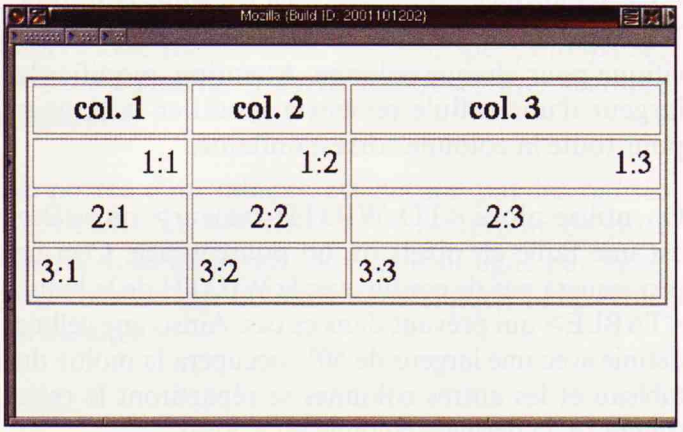
4) <TH>, dont nous n'avons pas encore parlé est un croisement entre <TD> et <TR>. TH signifie *Table*

Header (en-tête de tableau). Cette balise permet de donner un titre de colonne au tableau. Exemple :

```
<TABLE WIDTH="100%" BORDER="2" CELLSPACING="3"
CELLPADDING="5">

<TH>col. 1</TH><TH>col. 2</TH><TH>col. 3</TH>

<TR ALIGN="RIGHT">
 <TD>1:1</TD><TD>1:2</TD><TD WIDTH="50%">1:3</TD>
</TR>
<TR ALIGN="CENTER">
 <TD>2:1</TD><TD>2:2</TD><TD>2:3</TD>
</TR>
<TR ALIGN="LEFT">
 <TD>3:1</TD><TD>3:2</TD><TD>3:3</TD>
</TR>
</TABLE>
```

Notez que les balises <TH> s'utilisent en dehors d'une définition de ligne (<TR>). Le texte dans la portée de la balise est automatiquement en gras et centré dans la colonne (**figure 15**).

Fig.15

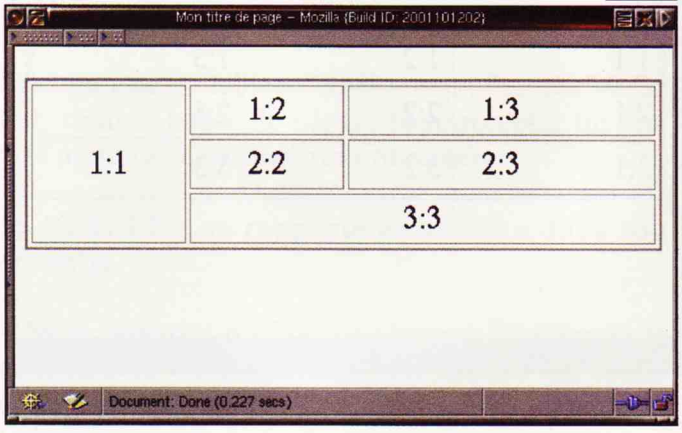


Deux autres options de la balise <TD> peuvent être utilisées. Nous avons choisi de les exposer à part en raison de leur implication sur l'ensemble d'un tableau. Il s'agit des paramètres ROWSPAN et COLSPAN qui permettent respectivement de fusionner des lignes et des colonnes. Voici un exemple concret :

```
<TABLE WIDTH="100%" BORDER="2" CELLSPACING="3"
CELLPADDING="5">
<TR ALIGN="CENTER">
 <TD ROWSPAN="3">1:1</TD><TD>1:2</TD><TD
WIDTH="50%">1:3</TD>
</TR>
<TR ALIGN="CENTER">
 <TD>2:2</TD><TD>2:3</TD>
</TR>
<TR ALIGN="CENTER">
 <TD COLSPAN="2">3:3</TD>
</TR>
</TABLE>
```

Notre tableau fait toujours trois colonnes et trois lignes. La cellule placée en première colonne de la première ligne utilise ROWSPAN pour s'étendre sur 3 lignes. De ce fait, les lignes qui suivent n'ont besoin que de deux cellules pour être complètes. La dernière cellule de la troisième ligne utilise COLSPAN pour s'étendre sur deux colonnes. Nous n'avons donc besoin que d'une paire de balises <TD></TD>. Le résultat en **figure 16** est heureusement plus parlant que les explications.

Fig.16



Les images

Il est possible en HTML d'intégrer des images dans ses pages. La balise utilisée est alors <IMG>. Celle-ci prend en argument un certain nombre de paramètres :

• SRC définit la source de l'image, c'est-à-dire le fichier graphique au format supporté par les navigateurs (PNG, JPEG ou GIF). On spécifiera alors l'emplacement sous une forme absolue par rapport à la racine du serveur avec <IMG SRC="/images/moi.png"> ou relativement au répertoire où est placé le fichier HTML en cours avec par exemple <IMG SRC=../images/moi.png"> (si nous sommes dans un répertoire de la racine du serveur).

• ALT permet d'afficher un commentaire en pop-up si le client laisse le pointeur de la souris suffi-

samment de temps sur l'image. Avec un client en mode texte, ce commentaire sera utilisé en lieu et place de l'image ne pouvant être affichée.

• WIDTH et HEIGHT déterminent la taille de l'image en pixels dans la page HTML. En l'absence de ces paramètres, l'image sera affichée telle quelle.

• BORDER donne un cadre à l'image. La valeur par défaut est 1 pixel si l'image est un lien (voir plus bas) et 0 dans le cas contraire.

• ALIGN permet d'aligner l'image sur le texte qui l'entoure. Par défaut, l'image est alignée sur le texte par le bas. Les arguments possibles sont LEFT, RIGHT, TOP, BOTTOM et MIDDLE (milieu).

• VSPACE et HSPACE permettent de définir une marge en pixels autour de l'image.

Les liens

HTML ne mériterait pas son nom sans le concept de liens hypertextes. Les liens permettent d'établir des liaisons entre du texte ou une image et une cible. Le point de départ d'un lien est défini par la portée des balises <A HREF=quelque chose> et </A>. N'importe quel élément présent dans cette portée constitue le point de départ du lien.

La cible spécifiée après le symbole = peut être une URL complète ou un nom de fichier. Exemple :

```
<A HREF="http://www.linux.org">Le site officiel de
Linux</A>
```

Ici, sur la page telle qu'elle apparaîtra au client, un clic sur la phrase "Le site officiel de Linux" renverra le navigateur vers le site **http://www.linux.org**. Il en va de même pour une image :

```
<A HREF="http://www.linux.org"><IMG
SRC="/images/pingouin.png" BORDER="0"></A>
```

Ici, c'est l'image affichée sans bordure qui sera le point de départ du lien.

```
<A HREF="/image/pingouin-gros.png"><IMG
SRC="/images/pingouin.png"</A>
```

Ici, nous ne renvoyons pas vers un site mais directement vers un fichier image au format PNG qui est présent dans notre propre arborescence du serveur HTTP. C'est habituellement la technique utilisée (en combinaison avec les tableaux) pour afficher des vignettes sur une page HTML et afficher une image en haute résolution si le visiteur clique dessus.

Un dernier type de lien vous permettra d'organiser les informations sur une grande page HTML. Imaginez que vous souhaitiez diffuser le présent article. En début de page, vous proposez un sommaire des différents points abordés et souhaitez lier ce sommaire avec un emplacement dans la page.

Il vous faudra tout d'abord définir des emplacements cibles avec :

```
<A NAME="tab">Les tableaux</A>
```

Dès lors, le titre "Les tableaux" devient une cible utilisable en argument de <A HREF=. Dans votre sommaire en début de page, vous ajouterez une ligne :

```
<A HREF="#tab">Infos sur les tableaux</A><br>
```

Le visiteur n'aura alors qu'à cliquer sur ce lien pour se retrouver automatiquement à l'emplacement "tab" sur la même page.

Conseil de rédaction

Terminons cette présentation de HTML avec un petit conseil qui vous évitera sans doute des tracas. La principale source d'erreur en HTML est l'oubli de fermeture des balises. Ainsi, sur un gros fichier, vous risquez de payer le prix fort pour avoir oublié un ou un quelque part dans le code.

La technique à adopter est la suivante : lorsque vous ouvrez une balise, refermez-la immédiatement et seulement ensuite, ajoutez le texte concerné. Vous vous assurerez ainsi qu'aucune balise n'est restée ouverte.

J'espère vous avoir donné goût à l'écriture manuelle de code HTML et que vous hésiterez à deux fois avant d'utiliser un quelconque générateur produisant du code souvent lourd et illisible...

PHP : le Web dynamique

PHP est un langage de programmation scriptable embarqué dans le HTML. En voilà un belle chose, me direz-vous ! Mais à quoi cela sert-il et en quoi cela peut-il m'être utile ? En pratiquant HTML durant une longue période, on se rend rapidement compte des limites imposées par un simple langage de description. PHP peut vous rendre des services quel que soit le type de site que vous souhaitez concevoir. Cela ira de la simple "page perso" au site de vente par correspondance. Nous allons dans cet article essayer de vous donner goût au langage PHP...

Nous venons de le voir dans l'introduction, la définition stricte d'un langage embarqué n'est pas très parlante. Nous allons donc dans un premier temps vous mettre l'eau à la bouche à l'aide un simple exemple. Soit le fichier HTML suivant :

```
<html>
<body bgcolor="#FFFFFF">
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
coucou<br>
</body>
</html>
```

Que de travail (même avec des copier/coller) pour écrire tout simplement dix fois "coucou" sur l'écran d'un navigateur. Et si nous voulions écrire 20 fois ce mot ? Et si nous voulions écrire un autre mot ? Il nous faudrait tout simplement modifier autant de lignes qu'il y a d'occurrences du mot.

Voilà à quoi ressemblerait ce même fichier en utilisant PHP :

```
<html>
<body bgcolor="#FFFFFF">
<?
for ($i=0;$i<10;$i++) {
  echo "coucou<br>\n";
}
?>
</body>
</html>
```

Sans entrer trop dans le détail, nous faisons ici une boucle. C'est une manipulation classique en termes de programmation. Il s'agit de fixer une valeur de départ à une variable (ici *$i*). Il nous suffit alors de "tourner" 10 fois dans la boucle pour afficher autant de fois le mot "coucou". En dehors du fait que le nombre de lignes est moindre avec le code que nous venons d'écrire, il faut prendre en considération le fait qu'il est extrêmement fonctionnel. Vous pouvez, en ne modifiant qu'une seule valeur, multiplier le nombre d'occurrences du mot "coucou" presque à l'infini (c'est tout à fait faisable mais absolument à éviter).

De la même manière, remplacer "coucou" par n'importe quel autre mot ou une phrase entière, ne vous prendra qu'une seconde.
J'espère que ce petit exemple vous aura convaincu de l'utilité et de l'intérêt de PHP et que vous poursuivrez la lecture du présent article.

PHP

Comme vous avez pu le voir, le PHP s'intègre directement dans le corps du HTML. L'insertion est marquée par les *tags* (balises) <? et ?>. Le support PHP doit être ajouté au serveur HTTP puisque c'est là que le code PHP sera transformé en HTML avant d'être envoyé au navigateur à l'autre bout d'Internet. L'installation du support PHP avec une distribution Debian est très simple, il vous suffira d'ajouter les packages adéquats.

```
# apt-get install php4
```

Dès lors, votre serveur Apache sera en mesure de traiter et d'interpréter le code PHP des pages. Dans votre fichier de configuration d'Apache, vous trouverez trace du support PHP grâce aux lignes qui y auront été ajoutées dans `/etc/apache/httpd.conf` :

```
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
```

qui charge le module PHP d'Apache, et dans `/etc/apache/mime.types` :

```
application/x-httpd-php                phtml pht php
```

qui permet d'associer les fichiers possédant une extension phtml, pht ou, plus couramment, php, avec l'interpréteur PHP. De cette manière, n'importe quel fichier portant l'extension *.php*, par exemple, passera par l'interpréteur PHP qui recherchera les tags <? et ?>. Tout le code se trouvant entre ces deux tags sera considéré comme des instructions PHP. Ces instructions, si elles ne comportent pas d'erreur, seront traduites en informations HTML et envoyées au client. Les fichiers ne comportant pas les tags <? et ?> passeront tout simplement au travers de l'interpréteur sans modification. Vous pouvez donc parfaitement utiliser un suffixe *.php* ou *.phtml* avec vos fichiers HTML classiques sans le moindre problème. Bien sûr, par l'intermédiaire de cette dernière ligne, il vous est possible de spécifier n'importe quel suffixe pour vos fichiers susceptibles de comporter du code PHP.

Vous pouvez par exemple utiliser un suffixe *.asp* correspondant au pendant de PHP dans le monde du logiciel propriétaire (notez au passage que ASP est également disponible sous GNU/Linux et sous une licence libre). Une autre solution est de tout simplement associer le suffixe *.html* avec le type *x-httpd-php* de manière à interpréter tout simplement tous vos fichiers aussi bien HTML que PHP. Attention cependant, ceci provoquera l'interprétation de tous vos fichiers, y compris ceux ne comportant pas de code PHP, ce qui provoquera une charge importante pour un système basé sur une "petite" configuration.

Normalement, l'installation du support PHP dans Apache a dû provoquer le redémarrage du service HTTP. Si tel n'est pas le cas, redémarrez manuellement le serveur avec :

```
# /etc/init.d/apache restart
```

Vous pourrez immédiatement vérifier le bon fonctionnement du support en créant un fichier `info.php` (par exemple) contenant le code suivant :

```
<? phpinfo(); ?>
```

Il s'agit d'une fonction PHP permettant d'afficher toutes les informations à disposition sur une page spécialement composée à cet effet (**figure 1**). Bien sûr, pour visualiser cette page, vous devrez pointer votre navigateur Web sur votre serveur HTTP et non sur le fichier lui-même. Rappelons que PHP est un langage embarqué interprété côté serveur et non côté client comme c'est le cas pour JavaScript. Cette simple fonction `phpinfo()` résume parfaitement le traitement de l'information. Les instructions PHP sont transformées en HTML (ici une simple fonction donne une page HTML complète) avant de l'envoyer au client.

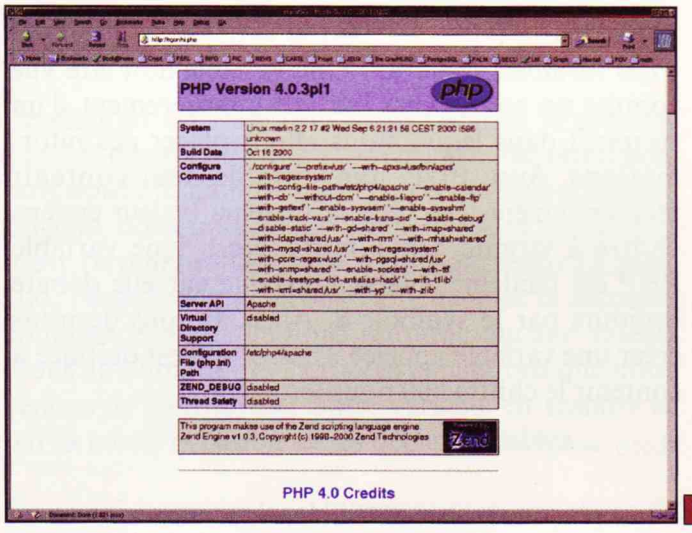


Fig.1

Attention ! Ne laissez pas ce fichier sur votre serveur HTTP, ou du moins ne le laissez pas accessible par tous. Il permet en effet d'obtenir des informations très sensibles (modules utilisés, version des logiciels, noms d'utilisateurs, etc.) sur le serveur.

Découverte du langage

PHP est un véritable langage de programmation. Syntaxiquement, il tient du Perl et du C. Il s'agit d'un langage interprété, c'est-à-dire que les fichiers écrits en langage PHP ne nécessitent pas de compilation ou de traitement spécifique de la part d'une application tierce. Ce que vous codez en PHP est

directement utilisable sur votre serveur Web.
PHP est un langage non typé. En clair, cela signifie, nous en ferons l'expérience plus loin dans l'article, qu'une variable peut contenir indifféremment une valeur numérique ou une chaîne de caractères. Par opposition, les langages typés comme le C obligent à définir une variable d'un certain type en fonction de ce qu'elle va contenir. Une variable pour un type numérique entier ne pourra pas être utilisée en C avec des fonctions qui traitent des chaînes de caractères.

Le non typage de PHP offre une souplesse mais peut également vous induire en erreur si vous tentez de multiplier le mot "toto" par 5, par exemple.
Les indications que nous allons donner à présent s'appliquent à un grand nombre de langages. Si vous êtes déjà programmeur, il est fort probable que cette section vous fasse mourir d'ennui. Cependant, dans le doute, jetez-y un œil, on ne sait jamais.
En premier lieu, il est nécessaire de bien définir le patois du programmeur...

Variables

- *Les variables scalaires* : Une variable doit être vue comme un contenant. Il s'agit grossièrement d'un récipient dans lequel nous allons placer des informations. Avec PHP, une variable peut contenir indifféremment tout type de contenu (valeur entière, chiffre à virgule, texte, lettre, etc.). Une variable PHP est facilement reconnaissable car elle débute toujours par le symbole $. Ainsi, si nous désirons créer une variable appelée `$toto` qui est destinée à contenir le chiffre 500 nous ferons :

```
$toto = 500;
```

Nous plaçons la valeur 500 dans `$toto`.

Remarque : La fin de la ligne est marquée par un point-virgule. Il est important de ne pas oublier cela. Ce symbole marque la fin de la ligne. Il est nécessaire car l'alignement du code PHP et la manière de disposer ce code en ligne n'a pas d'influence sur le comportement du code. Ainsi, si nous avions dû définir deux variables `$toto` et `$titi` devant contenir respectivement les valeurs 500 et 1000, nous aurions pu faire :

```
$toto = 500;
$titi = 1000;
```

mais aussi :

```
$toto = 500; $titi=1000;
```

Dès lors que nos variables sont définies et initialisées (on leur a donné un contenu), nous pouvons les manipuler comme s'il s'agissait des valeurs elles-mêmes :

```
$total = $toto + $titi;
```

`$total` contiendra alors la valeur 1500. Pour vous en convaincre, voici le code complet :

```
<html>
<body>
<?
 $toto = 500;
 $titi = 1000;
 $total = $toto + $titi;
 echo "Total : $total<br>";
?>
</body>
</html>
```

Nous verrons s'afficher sur l'écran de notre navigateur : *Total : 1500.*

Nous l'avons dit, une variable peut contenir aussi bien du numérique que du texte. Soit le code suivant (à partir de maintenant, nous assumerons le fait que vous ajouterez vous-même les balises HTML adéquates et les marqueurs <? et ?>) :

```
$toto = 500;
$titi = "coucou";
echo "$toto<br>$titi<p>";
$total = $toto + $titi;
echo "Total : $total<br>";
```

Résultat sur le navigateur :

500
coucou

Total : 500

Distinguez-vous le problème ? Nous avons tenté de faire la somme (une opération arithmétique) d'un nombre et d'un mot. Rappelez-vous de vos cours de primaire, "on ne peut pas additionner des choux et des carottes", pourquoi PHP le pourrait ? PHP ne signale pas d'erreur et se borne à faire ce qu'il peut pour vous satisfaire. Ici, il ne peut pas, bien sûr, additionner 500 et "coucou" et limite l'opération

arithmétique à ce qu'il peut faire : *500 + rien = 500*.

D'où l'intérêt de bien nommer ses variables. Utiliser *$toto* et *$titi* est stupide. Le problème ne se poserait pas avec :

```
$montant = 500;
$mot = "coucou";
```

Cela peut paraître insignifiant, mais c'est pourtant l'une des principales sources d'erreur en PHP. Apportez donc la plus grande attention à la manière de nommer vos variables et ce, le plus tôt possible.
Les variables que nous venons de voir sont des variables scalaires. Elles permettent de contenir des éléments simples. Mais il existe deux autres sortes de variables...

- Les tableaux. Un tableau en termes de programmation peut être à plusieurs dimensions. Commençons doucement avec un simple tableau à une seule dimension :

```
$prix[0]=100;
$prix[1]=140;
$prix[2]=120;
$prix[3]=10;
$prix[4]=75;
```

Ne vous y trompez pas, il s'agit bien d'une seule variable `$prix`. Mais comme il s'agit d'un tableau, on utilise un paramètre supplémentaire permettant de désigner la cellule de ce dernier. Là encore, une règle importante pour tout programmeur est de respecter la manière dont fonctionne un ordinateur. Eh oui, ce gros tas de circuits commence à compter à partir de 0 et non de 1 comme nous, pauvres humains. La première cellule est donc la cellule 0 et la dernière la numéro 4 (qui est la cinquième).

Nous pouvons ensuite procéder aux mêmes manipulations que sur les variables scalaires :

```
$prix[2]=$prix[0]+$prix[1];
```

`$prix[2]` contiendra alors la valeur 240. Nous avons additionné les valeurs des cellules 1 et 2 et placé la somme dans la première cellule.

Attention ! Toutes les cellules de `$prix` forment une seule variable. Cela signifie qu'à cause d'une simple erreur d'inattention, vous pouvez écraser entièrement votre tableau à l'aide d'une simple instruction comme `$prix=300;` A ce moment là, la variable `$prix` devient scalaire et est initialisée avec le nombre 300. Pour vous en convaincre, essayez le code suivant :

```
$prix[0]=100;
$prix[1]=140;
$prix[2]=120;
$prix[3]=10;
$prix[4]=75;

$autreprix=$prix;

echo "$autreprix[0]<br>$autreprix[1]<br>$autre-
prix[2]<br><br>";
echo "$prix[0]<br>$prix[1]<br>$prix[2]<br><br>";

$prix=500;
$autreprix[0]=1000;

echo "$autreprix[0]<br>$autreprix[1]<br>$autre-
prix[2]<br><br>";
echo "$prix[0]<br>$prix[1]<br>$prix[2]<br><br>";

echo "$prix<br>"
```

Le résultat est flagrant. Nous copions le tableau `$prix` dans `$autreprix` en guise de référence, puis nous affichons le contenu de quelques cellules. Ensuite, nous initialisons `$prix` avec la valeur 500 (c'est l'erreur) et la cellule 0 de `$autreprix` avec 100. Nous recommençons l'affichage et la seconde commande `echo` ne nous retourne aucune valeur. Nous affichons alors `$prix` et constatons que nous venons de transformer notre variable en scalaire et par la même occasion avons détruit le tableau.

En ce qui concerne les tableaux multidimensionnels, il suffit de multiplier les références entre crochets. Ainsi `$prix[0][0]` désignera la toute première cellule d'un tableau à deux dimensions et `$prix[0][0][0]`, la toute première d'un tableau à trois dimensions. Jusqu'ici, la représentation est aisée, deux dimensions étant un tableau classique et trois dimensions une sorte d'empilement de case en volume. Là où cela devient nettement plus drôle, c'est en utilisant encore plus de dimensions. La variable `$prix[0][0][0][0]` fait référence à une cellule d'un tableau à cinq dimensions. Ne cherchez pas, il est impossible de matérialiser mentalement un tel tableau alors qu'il est parfaitement utilisable en programmation. Il est cependant très rare

d'avoir à faire appel à ce genre de "monstruosité".

En tentant d'utiliser `echo` comme nous l'avons fait jusqu'à présent, vous allez rencontrer un problème. Cette instruction tolère en effet l'utilisation des variables dans la chaîne de caractère à utiliser mais seulement dans une certaine mesure. Avec `$prix[0][0]`, c'en est trop est vous obtiendrez "*Array[0]*" comme seule réponse sur le navigateur. Le problème vient du fait que la limite d'`echo` est `$prix[0]` et la valeur ainsi demandée n'existe pas.

Il faut alors utiliser l'instruction de concaténation qui est représentée par un point. Ce qui nous donne :

```
echo $prix[0][0]."<br>";
```

Ce point permet tout simplement de réunir variable et chaîne de caractères en une seule entité le temps de l'affichage. Ce type de manipulation est également valide pour l'initialisation de variable :

```
$phrase = "Le prix est de ".$prix[0][0]."
Euros.<br>";
echo $phrase;
```

- Les tables de hachage (*hash*) ressemblent beaucoup aux tableaux, si ce n'est qu'on fait référence à une cellule par un mot clef. Ainsi, si nous devons stocker les âges de différentes personnes, plutôt que de créer un tableau où les personnes seront référencées par un nombre, nous utiliserons une table de hachage de cette manière :

```
$personne{"carole"}=20;
$personne{"alice"}=32;
$personne{"marie"}=25;
$personne{"maurice"}=43;
```

Il est beaucoup plus aisé ensuite de réutiliser ces noms et de récupérer l'âge de chaque personne :

```
echo $personne{"carole"}."<br>";
```

Nous avons vu qu'il était facile de modifier le contenu d'une variable ou de l'écraser avec une nouvelle valeur. Il existe d'autres manières de le faire : l'incrémentation. Imaginons que notre table de hachage, que nous venons de présenter, soit dépassée de deux années. Nous devons dans le principe ajouter 2 à chaque contenu. Il existe un moyen très simple de le faire et ce, en une seule instruction :

```
$personne{"carole"}=$personne{"carole"}+2;
echo $personne{"carole"}." ans<br>";
```

$personne{"carole"} contiendra maintenant 22. Pour une incrémentation de 1, la syntaxe est encore plus simple :

```
$personne{"carole"}++;
echo $personne{"carole"}." ans<br>";
```

La décrémentation se fait de la même manière :

```
$personne{"carole"}=20;
echo $personne{"carole"}." ans<br>";
$personne{"carole"}=$personne{"carole"}-2;
echo $personne{"carole"}." ans<br>";
$personne{"carole"}--;
echo $personne{"carole"}." ans<br>";
```

Les conditions ou structures de contrôle

Attribuer des valeurs à des variables, les modifier et les afficher est une chose intéressante. Mais, entre nous, si le langage se limitait à cela, il n'y aurait pas vraiment d'intérêt concret. Il est heureusement possible d'aller plus loin et de comparer le contenu d'une variable à une valeur fixe pour réagir en conséquence. On impose donc une condition et en fonction du fait que celle-ci soit vérifiée (exacte) ou non, on détermine un code à exécuter. Nous allons voir maintenant quelques conditions de base :

- *if* (si). C'est sans doute la condition la plus simple. Il s'agit tout bonnement de vérifier si la condition est vraie ou non. En guise d'exemple, nous définissons une variable scalaire `$age` et l'initialisons avec une valeur :

```
$age = 20;
```

La condition que nous allons établir est tout simplement l'âge de majorité légale. Si `$age` contient un nombre d'années suffisant, nous afficherons un message en conséquence. Voici le code :

```
if($age>=18) {
 echo "Vous êtes majeur<br>;
}
```

La condition est fixée par l'instruction `if` et, entre parenthèses, la condition elle-même. Ce qui nous donne en français "Si `$age` est supérieure ou égale

à 18 alors..."
Entre accolades, nous plaçons le code à exécuter si la condition est vérifiée. Essayez ces lignes en changeant la valeur de `$age`.

Une instruction spécifique à `if` peut être ajoutée. Elle détermine le comportement à adopter si la condition n'est pas vérifiée :

```
if($age>=18) {
 echo "Vous êtes majeur<br>";
 } else {
 echo "Vous n'êtes PAS majeur<br>";
}
```

En français : "Si `$age` est supérieure ou égale à 18 alors on affiche majeur, sinon on affiche pas majeur".
Les conditions utilisables sont :

> supérieur
< inférieur
== égal
> supérieur ou égal
< inférieur ou égal
!= différent

- *while* (tant que) est un type de condition permettant de boucler sur le code entre accolades, tant que la condition est vérifiée. Attention, c'est là une des principales sources d'erreur chez les débutants : il ne faut pas oublier de changer la variable de la condition dans la portée de cette dernière. En cas contraire, votre boucle ne finira jamais (du moins, tant que le navigateur ne sera pas planté).

Cette condition s'utilise de la manière suivante :

```
$compteur=20;

while($compteur>0) {
 echo "Le compteur est à $compteur<br>";
 $compteur--;
}

echo "c'est fini ! compteur à $compteur.<br>";
```

Que se passe-t-il ? Nous commençons par initialiser la variable `$compteur` avec le nombre 20. La condition `while` est ensuite testée. Au premier passage, `$compteur` est supérieure à 0, nous exécutons donc le code entre accolades : nous affichons une phrase et la valeur de la variable, puis nous décrémentons `$compteur` de 1. En sortie du premier passage, `$compteur` fait alors 19. La condition est toujours vraie (19>0) ; on fait alors un second passage dans la boucle. Au cours du 20ième passage, l'instruction `$compteur--;` place `$compteur` à 0. La condition n'est plus vérifiée et on sort de la boucle. On arrive alors sur la dernière ligne.

- *do while* (faire tant que) est une variante de `while` :

```
$compteur=20;

do {
 echo "Le compteur est à $compteur<br>";
 $compteur--;
} while($compteur>0);

echo "c'est fini ! compteur à $compteur.<br>";
```

- *for* est un type de condition généralement utilisée pour les répétitions. L'avantage est de ne pas avoir à incrémenter ou décrémenter une variable dans la portée de la condition. C'est ce type de condition qui a été utilisé en tout début d'article pour tenter de vous convaincre de l'intérêt de PHP (j'espère que cela est réussi !) :

```
for ($i=0;$i<10;$i++) {
  echo "coucou<br>\n";
}
```

Ici, tous les éléments de la condition et du contrôle de cette dernière sont regroupés entre les parenthèses. On commence par définir la valeur de départ d'une variable (`$i=0`), la condition de la boucle (`$i<10`), et la modification de la variable (`$i++`). PHP s'occupe du reste et vous n'avez qu'à placer votre code dans la portée de la condition.

- *foreach* s'applique aux tableaux et aux tables de hachage. Cette condition permet de parcourir tous les éléments de ce type de variable :

```
$prix[0]=100;
$prix[1]=140;
$prix[2]=120;
$prix[3]=10;
$prix[4]=75;
foreach ($prix as $pcourant) {
  echo "$pcourant<br>\n";
}
```

Nous définissons un tableau à une dimension

contenant quelques prix. Nous utilisons ensuite `foreach` ainsi : "pour chaque élément de `$prix`, on crée `$pcourant` qui contiendra la valeur de la cellule courante". Ainsi, nous récupérons successivement le contenu de chaque cellule et dans la portée de la condition, nous pouvons utiliser cette dernière, qui est stockée dans `$pcourant`. La boucle se termine lorsqu'il n'y a plus de cellule dont extraire le contenu.

Pour les tableaux multidimensionnels, il est nécessaire d'imbriquer les conditions :

```
$prix[0][0]=100;
$prix[0][1]=140;
$prix[0][2]=120;
$prix[1][0]=10;
$prix[1][1]=14;
$prix[1][2]=12;

foreach ($prix as $niv1) {
  foreach ($niv1 as $niv2) {
    echo "$niv2<br>\n";
  }
}
```

Il vous faudra autant de `foreach` que de dimensions dans votre tableau.

`foreach` s'applique également aux tables de hachage :

```
$personne{"carole"}=20;
$personne{"alice"}=32;
$personne{"marie"}=25;
$personne{"maurice"}=43;

foreach ($personne as $clef => $valeur) {
  echo "$clef a $valeur ans.<br>";
}
```

Ici, `foreach` extrait la clef et la valeur de chaque cellule/champ et les place dans des variables scalaires `$clef` et `$valeur` que nous pouvons utiliser dans la portée de la condition. Nous obtenons donc dans la fenêtre du navigateur :

carole a 20 ans.
alice a 32 ans.
marie a 25 ans.
maurice a 43 ans.

- *switch* est une condition très particulière, puisqu'elle permet de dispatcher le comportement à adopter en plusieurs morceaux de code. On teste alors une condition d'égalité. Si, par exemple, vous désirez afficher un message pour les personnes dont l'âge est 18 ans, un autre pour celles qui en ont 60, encore un autre pour les 20 ans, et un dernier message pour toutes les autres, vous utiliserez :

```
$age=20;

switch ($age) {
  case 18:
    echo "Vous êtes tout juste majeur.<br>";
    break;
  case 20:
    echo "C'est beau d'avoir 20 ans !<br>";
    break;
  case 60:
    echo "Alors, la retraite ?<br>";
    break;
  default:
    echo "Vous avez $age ans.<br>";
}
```

On définit la condition sur le contenu de la variable `$age` (on *switche* sur `$age`). Suivent ensuite les différentes valeurs que nous souhaitons évaluer et nous définissons un message à afficher en conséquence (`case` = dans le cas où `$age` est...)
Enfin, on précise un message si aucun cas défini ne s'est présenté (`default`). `break` marque la fin du test et la sortie de la condition. Essayez de retirer cette commande, vous allez vite comprendre...

Les fonctions

Nous avons eu un aperçu des fonctions avec `phpinfo();`. Une fonction est une instruction permettant de traiter des informations ou de générer des informations en une seule commande. `phpinfo()`, par exemple, est la fonction qui permet de générer une page HTML complète regroupant les informations sur le serveur HTTP, le système et le langage PHP installé.

Il existe un très grand nombre de fonctions avec PHP. Une fonction peut prendre en argument des paramètres. Ces derniers seront utilisés dans la fonction pour générer un résultat.
En dehors des fonctions définies par le langage lui-même, le développeur PHP a la possibilité de développer ses propres fonctions. C'est d'ailleurs la manière la plus simple de comprendre comment ce système fonctionne. Le fait de développer vos

propres fonctions vous permettra d'organiser votre code et de le rendre plus lisible. Voici un premier exemple :

```
function multiplication() {
  $resultat=10*5;
  return($resultat);
}
```

Nous définissons une nouvelle fonction appelée `multiplication` ; celle-ci ne prend en compte aucun paramètre, d'où l'absence d'élément entre parenthèses. La portée de la fonction est définie par les accolades. Ainsi, nous créons une variable `$resultat` contenant le produit de 10 par 5. Enfin, nous spécifions ce que doit retourner la fonction avec `return`. La variable `$resultat` n'est définie que dans la portée de la fonction. Elle n'a aucune existence en dehors.

Nous pouvons ensuite, dans le reste du code, utiliser notre belle fonction de cette manière :

```
$monresultat=multiplication();

echo "$monresultat<br>";
echo "$resultat<br>";
```

`$monresultat` est initialisé avec ce que retourne `multiplication()`. Il s'agit ici d'une valeur, mais une fonction peut retourner n'importe quel type de variables (scalaire, tableau, table de hachage) et de contenu (numérique entier, texte, etc.).

Nous affichons le contenu de `$monresultat` et tentons également d'afficher celui de `$resultat`. On constate alors que `$monresultat` contient effectivement le produit calculé par la fonction mais que `$resultat` n'est pas défini.

Poussons plus loin et transformons notre fonction de manière à ce qu'elle accepte un argument :

```
function multiplication($nombre) {
  $resultat=$nombre*5;
  return($resultat);
}
```

Cette fois, le paramètre passé lors de l'utilisation de la fonction sera automatiquement placé dans `$nombre`. Nous utilisons ensuite cette variable (qui n'existe que dans la portée de la fonction) pour calculer notre produit. Nous retournons le résultat comme précédemment. Voici comment utiliser cette fonction :

```
monresultat=multiplication(6);
echo "$monresultat<br>";
```

Et voilà, le navigateur nous affiche tout naturellement 30...

Multiplier les paramètres n'est pas un problème, il nous suffit de prévoir plusieurs variables pour les stocker et pour procéder au calcul du produit :

```
function multiplication($nombre,$multiplicateur) {
  $resultat=$nombre*$multiplicateur;
  return($resultat);
}

$monresultat=multiplication(6,4);
echo "$monresultat<br>";
```

Les problèmes arrivent lorsque le nombre de paramètres attendus et spécifiés est différent. PHP avertira l'utilisateur à l'aide d'un message mais n'arrêtera pas le script :

Warning: Missing argument 2 for multiplication( ) in /var/www/base.php on line 5
0

Pour faire propre, nous devons gérer cela à l'intérieur de notre fonction :

```
function multiplication($nombre,$multiplicateur) {
  if(!$nombre or !$multiplicateur) {
    echo "<b><font color=\"#FF0000\">Erreur, tous
les arguments ne sont pas présent
!</b></font><br>";
    exit;
  }
  $resultat=$nombre*$multiplicateur;
  return($resultat);
}
```

Nous avons ajouté une condition `if` permettant de vérifier la présence des deux variables à utiliser. Nous en profitons pour introduire deux nouvelles notions :

- en spécifiant simplement le nom d'une variable dans la condition, nous vérifions son existence. Ainsi, `if($variable)` est vérifié si `$variable` est initialisé avec une valeur et `if(!$variable)` est vérifié si la variable est indéfinie (n'a jamais été

initialisée et donc contient une valeur non définie).

• une opération logique peut être utilisée pour spécifier plusieurs conditions dans un même test. Ici, nous utilisons `or` pour notre condition : "la condition est vérifiée si `$nombre` **OU** `$multiplicateur` est indéfini". Un autre opérateur logique pourrait être `and` pour ET. Voici à quoi ressemblerait notre fonction avec cet opérateur :

Remarque : On notera qu'en respectant les standards HTML, la définition d'une couleur doit être placée entre guillemets. Or, les guillemets sont utilisés avec PHP pour définir une chaîne de caractères. Nous devons donc spécifier que les guillemets qui concernent le code HTML ne sont pas à prendre en compte pour le code PHP, mais font partie intégrante de la chaîne de caractères à afficher. Nous plaçons donc un anti-slash devant les occurrences en question. Ceci est couramment appelé "échapper des caractères".

```
function multiplication($nombre,$multiplicateur) {
  if($nombre and $multiplicateur) {
    $resultat=$nombre*$multiplicateur;
    return($resultat);
  } else {
    echo "<b><font color=\"#FF0000\">Erreur, tous
les arguments ne sont pas présent
!</b></font><br>";
    exit;
  }
}
```

"Si `$nombre` **ET** `$multiplicateur` sont définis, on procède au calcul, **SINON** on affiche un message d'erreur et on stoppe (*exit*) l'interprétation du code".

PHP et HTML

Nous avons implicitement signalé que tout ce qui est affiché à l'aide de l'instruction `echo` figurera dans le code HTML final. C'est là toute la base du fonctionnement de PHP comme langage embarqué dans HTML et également sa grande force par rapport à l'utilisation de scripts ou d'exécutables CGI.

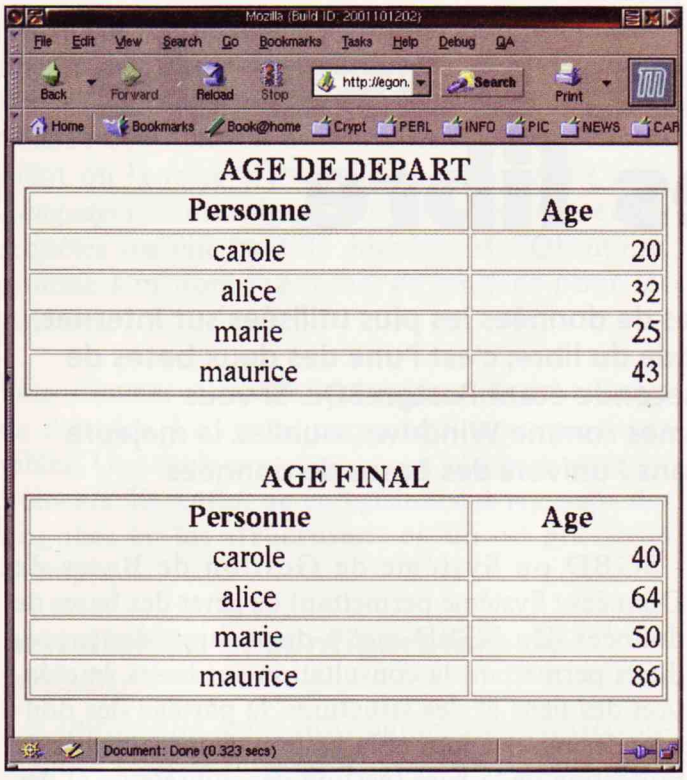
L'imbrication de PHP dans HTML permet ainsi de créer des fichiers mixtes comme celui listé en début d'article. On comprend alors pleinement les avantages de PHP. Voici un bel exemple regroupant quelques notions que nous venons d'apprendre :

```
<html>
<body>
<?
function multiplication($hash,$multiplicateur) {
  if(!$hash or !$multiplicateur) {
    echo "<b><font color=\"#FF0000\">Erreur, tous
les arguments ne sont pas présent
!</b></font><br>";
    exit;
  }

  foreach ($hash as $clef => $valeur) {
    $hash{$clef}=$valeur*$multiplicateur;
  }

  return($hash);
}

$personne{"carole"}=20;
$personne{"alice"}=32;
$personne{"marie"}=25;
$personne{"maurice"}=43;

$doublepers=multiplication($personne, 2);

echo "<center><b>AGE DE DEPART</b><br>";
echo "<table border=2 width=\"100%\">";
echo "<th>Personne</th><th>Age</th>";

foreach ($personne as $clef => $valeur) {
  echo "<tr><td align=center>$clef</td><td
align=right>$valeur</td></tr>";
}
echo "</table><br><br>";

echo "<center><b>AGE FINAL</b><br>";
echo "<table border=2 width=\"100%\">";
echo "<th>Personne</th><th>Age</th>";
foreach ($doublepers as $clef => $valeur) {
  echo "<tr><td align=center>$clef</td><td
align=right>$valeur</td></tr>";
}
echo "</table>";

?>
</body>
</html>
```

Je vous laisse, en guise d'exercice, le soin d'autopsier ce code, qui fournira le résultat donné en figure 2. Voici, en bonus, une manière de factoriser davantage ce code en créant une fonction d'affichage des tableaux :

```
function afftab($titre,$tableau) {
```

Fig.2




```
  echo "<center><b>$titre</b><br>";
  echo "<table border=2 width=\"100%\">";
  echo "<th>Personne</th><th>Age</th>";

  foreach ($tableau as $clef => $valeur) {
    echo "<tr><td align=center>$clef</td><td
align=right>$valeur</td></tr>";
  }
  echo "</table><br>";
}
```

et pour l'utiliser :

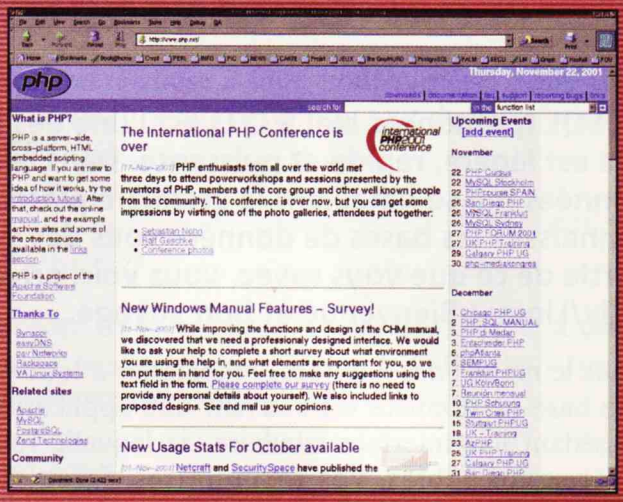
```
afftab("AGE DE DEPART",$personne);
afftab("AGE FINAL",$doublepers);
```

Nous arrivons au terme de cet article. Nous n'avons, de loin pas, couvert tous les aspects du langage PHP (ce qui aurait pris plusieurs magazines complets au bas mot), mais nous avons normalement fourni les bases qui vous permettront de débuter et de créer vos premières pages.

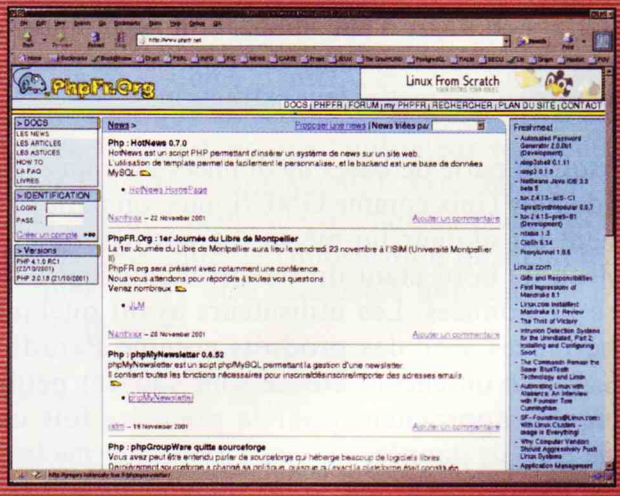
Il existe un grand nombre de sources d'information à propos de PHP, à commencer par les sites s'y rapportant mais aussi et surtout les ouvrages spécifiques. Vous trouverez dans le bloc de liens un certain nombre de sources vous permettant de poursuivre votre apprentissage.

Liens

PHP homepage
http://www.php.net



PHP en français :
http://www.phpfr.org/



Livres :

Programmation en PHP (nouvelle édition)
Leon Atkinson
Editions CampusPress

Programmation Web avec PHP
L. Lacroix
Editions Eyrolles

Professional PHP Programming
J. Castagnetto, H. Rawat, S. Schumann, C. Scollo, D. Veliath
Editions Wrox

MySQL : une base de données libre

MySQL (prononcez maï S Q L) est l'une des bases de données les plus utilisées sur Internet. Elle est légère, rapide et puissante. Dans le monde du libre, c'est l'une des deux bases de données qui se bataillent la première place, la seconde étant PostgreSQL. Si vous connaissez les bases de données sous des systèmes comme Windows, oubliez la majeure partie de ce que vous savez. Vous voici arrivé dans l'univers des bases de données Unix/Linux ! Bienvenue et bon voyage...

Dans le monde des interfaces graphiques à tout va, une base de données est souvent une application possédant une interface tabulaire via laquelle vous créerez vos tables, vos formulaires, ferez des requêtes, des états et tout un tas d'autres choses. En réalité, une base de données n'est pas du tout ça. La base elle-même est le moteur qui se cache derrière l'interface. Il s'agit d'une application accomplissant une seule tâche : regrouper des données de manière structurée et permettre leur extraction.

Lorsqu'on parle de bases de données avec un système de type Unix comme GNU/Linux, on parle réellement du moteur lui-même. La carrosserie et le tableau de bord étant du ressort d'un client de la base de données. Les utilisateurs ayant quelques expériences avec des produits comme Paradox, MS/Access ou encore DBase sont souvent perdus lorsqu'ils approchent pour la première fois une vraie base de données au sens strict du terme (et je parle d'expérience). Ne vous attendez donc pas à une application graphique, des petits boutons partout et de belles fenêtres. Une bonne base de données fait correctement son travail et ne doit pas être jugée de par son aspect. Vous voilà prévenu...

Rappel sur les bases de données

Pour partir sur une bonne base, rien de tel que quelques définitions :

• Base de données : Ensemble de données dont la structure reflète l'existence de relations entre ces données. En moins académique, il s'agit tout simplement d'un ensemble de données réunies de manière structurée avec des liens entre les différentes informations.

- SGBD ou Système de Gestion de Bases de Données : Système permettant de gérer des bases de données. Un SGBD met à disposition des procédures permettant la consultation des bases, la création des liens et des structures, le partage des données, les mises à jour et la gestion de mécanismes de protection de la base (gestion des utilisateurs et des accès).

Un SGBD doit assurer également l'indépendance des données vis-à-vis du matériel. C'est un point très important car du point de vue de l'utilisateur, la seule relation qu'il peut avoir avec les données se fait par l'intermédiaire du SGBD. C'est ici que la fraction s'opère si vous avez l'habitude d'utiliser des SGBD Windows (entre autres). Avec MySQL, vous n'avez pas le droit (et si vous êtes root, vous n'avez pas intérêt) à toucher directement aux fichiers de données gérés par MySQL.

• Tables, Enregistrement et Champs : Base de données structurée sous forme de tables. Vous pouvez vous représenter une table comme un fichier (au sens bureautique/mobilier du terme). Dans un fichier, on place des fiches, ce sont les enregistrements. Ces enregistrements possèdent une structure fixe et prédéfinie (petites cases et zones sur la fiche), ce sont les champs.

Ainsi, on ne placera pas, par exemple, un fichier client dans le fichier des articles. De la même façon, on ne trouvera pas de champ concernant un article sur la fiche d'un client. Bien que dans le monde réel vous puissiez faire ce genre d'erreur (au risque d'une sévère réprimande), ceci est formellement interdit avec un SGBD et refusé par le système.

• Requête : Ordre donné au SGBD afin qu'il exécute une opération sur une base de données. On distinguera plusieurs types de requêtes pour la consultation, l'enregistrement de nouvelles données, l'effacement ou la mise à jour. SQL (*Structured Query Language*) est un langage normalisé pour lancer des requêtes sur une base de données. MySQL utilise, comme son nom l'indique, ce langage pour les requêtes.

Pour résumer : un SGBD peut gérer plusieurs bases de données. Une base regroupe une ou plusieurs tables. Une table regroupe un ou plusieurs enregistrements. Et, enfin, un enregistrement regroupe des données en les structurant en un ou plusieurs champ(s).

Installation et configuration de MySQL

Les utilisateurs d'une distribution Debian (recommandée pour la mise en œuvre d'un serveur Web) sont décidément des veinards. L'installation de MySQL se limitera à l'utilisation d'une seule commande (en tant que root) :

```
# apt-get install mysql-server
```

Cette commande aura pour effet d'installer le serveur, mais également le client standard MySQL et de mettre en place une configuration minimale. Si vous utilisez une autre distribution, reportez-vous au manuel de cette dernière et à la documentation de MySQL, sans doute présente dans `/usr/doc` ou `/usr/share/doc`. Les instructions concernant la configuration et les fichiers de configuration qui vont suivre se rapportent toutes à une distribution Debian. Il ne devrait cependant pas y avoir de gros changements pour appliquer MySQL à d'autres distributions connues.

La première étape est de sécuriser un minimum l'installation par défaut. Pour l'heure, n'importe quel utilisateur a accès au serveur MySQL et surtout, n'importe quel utilisateur peut prendre l'identité root sur le serveur.

Attention ! Le super utilisateur root de la base de données n'est pas le même que celui du système. Cette désignation est source de confusion, mais en un sens justifiée. Il y a un root pour le système et un AUTRE root pour le serveur MySQL. Le root MySQL est le super-DBA, c'est-à-dire l'autorité la plus puissante du système de gestion de bases de données. Pour éviter les confusions dans cet article, nous parlerons de l'utilisateur root de MySQL en utilisent root/MySQL et de l'utilisateur root du système en utilisant simplement root.

La première chose à faire pour protéger votre serveur MySQL est de définir un mot de passe pour le root/MySQL :

```
/usr/bin/mysqladmin -hlocalhost -uroot password
'mot_de_passe'
```

Procédez de même avec `-hnommachine`. Suite à cela, dans le répertoire `/root` (celui de l'utilisateur root), vous devrez créer un `.my.cnf` contenant :

```
[mysqladmin]
user            = root
password        = mot_de_passe
```

Ceci est impératif car l'utilisateur root est le seul à pouvoir lancer, stopper ou redémarrer le service `mysql` via le système d'init. C'est également sous l'identité root que les scripts du *scheduler* `cron` sont exécutés. Il faut donc que l'utilisateur root puisse utiliser l'identité root/MySQL sans bloquer les scripts par une demande de mot de passe. Etant donné la sensibilité de l'information qu'il contient, ce fichier DOIT être accessible uniquement par l'utilisateur root :

```
# chmod 0600 /root/.my.cnf
```

MySQL stocke les permissions et les privilèges des utilisateurs via une base de données appelée `mysql`. Celle-ci comporte une table `user` regroupant les identités et les privilèges de chaque utilisateur. Voilà pourquoi nous avons dû utiliser deux fois la commande `mysqladmin` car nous avions deux entrées root dans la table `user` (une par nom de machine `localhost` et `egon`).

Dès à présent, vous pourrez vous connecter au serveur MySQL avec le client `mysql`. Ce dernier prend en compte plusieurs arguments sur la ligne de commande :

• *hhote* pour spécifier l'hôte sur lequel réside le serveur MySQL. Par défaut `localhost` sera utilisé.
• *uutilisateur* pour spécifier l'identité à prendre sur

le serveur MySQL. Par défaut, l'identité de l'utilisateur système sera utilisée.
• *pmot_de_passe* permet de préciser le mot de passe correspondant à l'utilisateur défini. En l'absence de cette option et si l'utilisateur possède un mot de passe, la connexion sera refusée. Si seule l'option `-p` est présente et qu'aucun mot de passe n'est précisé, l'utilisateur sera invité à le saisir.
• *base* spécifie la base de données à laquelle se connecter. Ce paramètre est optionnel et vous pourrez, une fois le client en relation avec le serveur, vous connecter à une base avec `\u nom_base`.

En principe, l'utilisateur root/MySQL n'est pas censé travailler sur une base de données autre que `mysql`. Il nous faut donc créer un nouvel utilisateur sur le serveur MySQL. Nous choisirons ici `wwwdata` puisqu'il rappelle le nom d'utilisateur sous lequel fonctionne Apache.

La création d'un utilisateur et l'attribution de ses privilèges se fait en une seule et même étape. Pour cette manipulation, nous allons créer un nouvel enregistrement dans la table `user` de la base `mysql`. La commande GRANT permet de conférer des privilèges à un utilisateur.

Commençons par créer notre première base de données que nous appellerons `mabase`. Pour cela, lancez le client `mysql` en tant que root/MySQL :

```
$ mysql -uroot -pmot_de_passe
Welcome to the MySQL monitor.  Commands end with ;
or \g.
Your MySQL connection id is 95 to server version:
3.22.32-log

Type 'help' for help.

mysql>
```

Cette dernière ligne est l'invite du client `mysql`, c'est ici que vous saisirez vos requêtes SQL et les commandes destinées au serveur MySQL lui-même. Notre première commande consistera à créer `mabase` :

```
mysql> create database mabase;
Query OK, 1 row affected (0.00 sec)
```

Toutes les commandes que vous utiliserez dans le client `mysql` devront se terminer par ; (ou `\g`). Ceci peut paraître pénible au premier abord, mais en réalité cela permet de saisir des requêtes sur plusieurs lignes. De telles requêtes ne seront lancées que lorsqu'une ligne avec ; sera validée.

La seconde étape consiste à donner des privilèges à l'utilisateur `wwwdata` sur cette nouvelle base :

On se connecte d'abord sur `mabase` :

```
mysql> \u mabase
Database changed
```

Puis on accorde les privilèges sur toutes les tables de `mabase` :

```
mysql> GRANT ALL PRIVILEGES ON mabase.* TO wwwda-
ta@localhost;
Query OK, 0 rows affected (0.07 sec)
```

Nous pouvons à présent quitter le client `mysql` avec `\q`, `quit` ou encore <CTRL>-D, puis on donnera un mot de passe à l'utilisateur `wwwdata` :

```
$ mysqladmin -p -uwwwdata password 'mot_de_passe'
```

Enfin, il ne nous reste plus qu'à nous connecter à `mabase` sur le serveur en utilisant l'identité `wwwdata` et le mot de passe correspondant :

```
$ mysql -uwwwdata -pmot_de_passe mabase
Welcome to the MySQL monitor.  Commands end with ;
or \g.
Your MySQL connection id is 98 to server version:
3.22.32-log

Type 'help' for help.

mysql>
```

Nous sommes maintenant prêts à découvrir SQL...

Premiers pas

Notre base est fraîchement créée, nous utilisons une identité spécifique et nous avons réglé le problème des privilèges. Bref, matériellement, tout est prêt mais nous n'avons rien :)

La première étape consistera donc à créer une première table. En guise d'exemple, celle-ci se rapportera à un répertoire d'amis. Le sujet est suffisamment vaste pour explorer de nombreuses possibilités qu'offre MySQL. Nous allons stocker toutes les coordonnées de nos amis pour ensuite les utiliser au mieux et établir des liens avec d'autres informations.

Pour l'heure, penchons-nous sur les éléments de notre table de coordonnées. Avant de concevoir une base, il convient de bien réfléchir à la structure à adopter et à la manière dont nous allons répartir l'information. Le maître mot pour cela est "répétition". En effet, il n'y a pas de raison de répéter des informations statiques (dont les variantes sont limitées) qui seront communes à plusieurs personnes.

Quelles sont les informations répétitives dans les coordonnées d'une personne ? Autrement dit, sur un panel d'amis, quelle est ou quelles sont les informations qu'ils pourront avoir en commun ? La réponse est : la civilité. Cette information étant générale à tous nos amis, nous pouvons la stocker dans une table séparée. Ainsi, si un jour nous avons besoin d'une nouvelle civilité, il nous suffira d'ajouter un enregistrement dans la table en question.

Comme nous stockons cette information dans une table séparée, nous avons besoin d'une information permettant d'établir un lien entre cette table et nos amis. Cette information sera un code. Comme nous sommes des gens qui n'aimons pas perdre de temps en saisie inutile, il serait très agréable de faire en sorte que ce code (numérique) augmente tout seul au fil des ajouts dans la table.

Voilà, nous avons toutes les informations nécessaires. Dans notre table de civilité, nous aurons :
- un code qui s'auto-incrémente ;
- une civilité ;
- une civilité abrégée.

La seconde étape consiste à choisir un type de données pour chacune de ces informations, mais également une taille maximale. Ici, ce n'est pas très compliqué:

- *code* est une valeur numérique entière. En ce qui concerne sa taille, MySQL propose plusieurs choix :

TINYINT est une valeur entière entre -127 et 128. Il est possible de préciser que nous n'avons pas besoin de la valeur négative, ce qui augmente d'autant la valeur maximum. On parle alors d'entier non-signé. Un TINYINT non-signé permet de stocker une valeur entre 0 et 255
SMALLINT permet une valeur entre -32 768 et 32 767 signée et de 0 à 65 535 en non-signé.
MEDIUMINT permet de -8 388 608 à 8 388 607 en signé et de 0 à 16 777 215 en non-signé.
INT est l'entier classique en signé ; il permet de stocker une valeur entre -2 147 483 648 et 2 147 483 647. En non-signé, cela nous donne de 0 à 4 294 967 295.
BIGINT enfin est le plus gros des entiers ; il permet d'aller de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 en signé et de 0 à 18 446 744 073 709 551 615 en non-signé.

Il est très peu probable d'avoir affaire un jour à plus de 256 civilités (et encore moins à plus de 4 milliards). Un *TINYINT* nous sera donc suffisant.

- pour *civilité*, nous avons le choix entre *CHAR* qui est un type de données caractères de taille fixe et *VARCHAR* pour une taille variable. Une entrée dans un champ de type *CHAR* sera toujours complétée par des blancs pour obtenir la taille fixée. Pour *VARCHAR*, la taille étant variable, les blancs et les espaces en fin de chaîne de caractères sont supprimés avant ajout. C'est, bien sûr, le plus souvent *VARCHAR* qui sera choisi. En ce qui concerne la taille de *VARCHAR*, il s'agit d'un maximal entre 1 et 255 caractères. Pour la civilité complète, un maximal de 80 devrait être largement suffisant en comptant une marge de tolérance.

- enfin, la *civilité abrégée* sera aussi du type *VARCHAR* avec une taille maximale de 10. Permettre davantage de caractères n'autoriserait pas l'utilisation du qualificatif "abrégée".

Nous en avons fini avec la théorie, voyons la pratique. Voici tout d'abord la requête SQL à utiliser :

```
mysql> CREATE TABLE civil (
    ->   ccode TINYINT UNSIGNED DEFAULT '0' NOT
NULL auto_increment,
    ->   cciv VARCHAR(80),
    ->   cabreg VARCHAR(10),
    ->   PRIMARY KEY (ccode));
Query OK, 0 rows affected (0.00 sec)
```

La requête SQL est composée d'une clause (*CREATE TABLE*) suivie du nom de la table à créer et de la structure de cette dernière entre parenthèses. La première ligne de la définition de structure est importante car elle concerne le code. Nous précisons le nom du champ suivi de son type et des options souhaitées. Ici, la valeur par défaut du champ est 0, le champ ne peut pas être vide (*NOT NULL*) et on spécifie une auto-incrémentation. La dernière ligne définit un index pour ce champ. Ceci

est absolument nécessaire pour l'auto-incrémentation et permet d'autre part d'accélérer les recherches.

Les deux autres définitions de champs se passent d'explications supplémentaires. Il y a cependant une remarque importante à faire sur la manière de nommer les champs dans une table. Vous l'aurez vu ici, notre table `civil` ne comporte que des champs commençant par la lettre "c" et ce n'est pas un hasard. En effet, une base de données est souvent constituée de plusieurs tables (parfois jusqu'à plusieurs dizaines). Il nous faut donc au moins reconnaître efficacement à quelle table appartiennent les champs que nous utiliserons dans nos requêtes d'interrogation. Si nous prenons notre code de civilité et que nous le nommons `code`, il est fort probable que ce nom revienne dans d'autres tables (code d'événement, code d'activité sportive, etc.). Ceci ne pose pas de problème à MySQL puisqu'il est possible d'utiliser un même nom de champ dans plusieurs tables et de s'en servir en précisant `table.champ` dans nos requêtes (comme `civil.ccode` ici). Reconnaissez cependant qu'il est plus aisé d'utiliser `ccode` que `civil.code`.

La démarche en plusieurs étapes que nous avons adoptée peut paraître un peu... trop explicative. Mais c'est ainsi que vous devrez travailler. Avant de créer des tables, il faut mûrement réfléchir aux données que vous allez y placer. Modifier la structure d'une table contenant des données est possible mais absolument déconseillé.

Imaginez un instant que vous ayez à créer une base de données plus professionnelle permettant de stocker des informations relatives à la gestion de commandes ou de factures. En vous attaquant directement à la pratique, vous avez toutes les chances d'oublier des champs, de ne pas prévoir des situations particulières ou tout simplement de définir une taille trop petite pour certains champs. Si votre projet passe en production, et que vous vous rendiez compte de l'erreur au bout de quelques semaines ou quelques mois, la modification de la structure de la base ne sera peut-être plus du tout possible.

Vous vous serez mis presque délibérément dans une situation fâcheuse. Passez donc un temps conséquent à analyser vos besoins, dessiner vos tables sur papier, partager vos choix avec d'autres personnes et prévoir un maximum de situations. Cette étape est l'analyse. Cela est certes très déplaisant si vous êtes programmeur, mais elle n'en est pas moins indispensable. Sachez également qu'il existe des méthodes de travail standard pour faire cette analyse, la plus connue étant, par exemple, Merise.

Enchaînons directement avec la table contenant les coordonnées de nos amis. De quoi avons-nous besoin ?
- un code numérique auto-incrémenté
- une civilité
- un nom
- un prénom
- une première ligne d'adresse
- une seconde ligne d'adresse
- un code postal
- une ville
- un pays
- un numéro de téléphone
- une date de naissance

Nous ne recommencerons pas une analyse complète ici, le point nouveau que nous n'avons pas encore abordé est le problème de stockage d'une date. Une date est un type de données bien particulier, il fait partie des informations temporelles. MySQL gère plusieurs types de données de ce genre :

• *DATE* stocke des dates au format AAAA-MM-JJ de 1000-01-01 à 9999-12-31 (vous avez dit bogue Y2K ? :)

• *TIME* stocke une heure au format HH:MM:SS entre -838:59:59 et 838:59:59.

• *DATETIME* stocke à la fois une date et une heure au format AAAA-MM-JJ HH:MM:SS entre 1000-01-01 00:00:00 et 9999-12-31 23:59:59.

• *TIMESTAMP* permet de stocker une date et une heure au format *timestamp* Unix, c'est-à-dire un nombre de secondes écoulé depuis le début de l'époque (1970-01-01 00:00:00, premier janvier 1970 à minuit pile). Attention, la limite pour ce format est une date durant l'année 2037 (prévoyez large).

• *YEAR* stocke tout simplement une année entre 1901 et 2155.

Pour notre nouvelle table, nous utiliserons tout naturellement le type *DATE*. Voici notre requête SQL :

```
mysql> CREATE TABLE amis (
    ->   acode SMALLINT UNSIGNED DEFAULT '0' NOT
NULL auto_increment,
    ->   aciv TINYINT,
    ->   anom VARCHAR(80),
    ->   aprenom VARCHAR(80),
    ->   aadd1 VARCHAR(255),
    ->   aadd2 VARCHAR(255),
    ->   acp VARCHAR(8),
    ->   aville VARCHAR(80),
    ->   apays VARCHAR(80),
    ->   atel VARCHAR(20),
    ->   adate DATE,
    ->   PRIMARY KEY (acode));
Query OK, 0 rows affected (0.00 sec)
```

Note : Tout comme pour la table `civil`, nous avons nommé nos champs en les préfixant d'un caractère correspondant au premier caractère du nom de la table. Nous avons choisi un *SMALLINT* non signé pour `acode`. Il est très peu probable que nous ayons plus de 65535 amis ; dans le cas contraire, ouvrez un fan club :)

Insérer des données

Nous venons de créer deux tables, elles sont directement utilisables et nous pouvons y placer des données en respectant la limite que nous nous sommes fixée. C'est également l'occasion de voir l'une des commandes les plus intéressantes en ce qui concerne le travail sur des tables que nous n'aurions pas créées. Des commandes spécifiques à MySQL permettent en effet d'obtenir un certain nombre d'informations sur nos tables :

`show tables` nous permet de lister les tables présentes dans la base :

```
mysql> show tables;
+------------------+
| Tables in mabase |
+------------------+
| amis             |
| civil            |
+------------------+
2 rows in set (0.00 sec)
```

Nous retrouvons nos deux créations. Nous pouvons ensuite lister la structure de chacune des tables (par exemple, *amis*) :

Voir tableau ci-dessous.

Toutes les informations intéressantes s'affichent automatiquement et nous retrouvons la définition complète de la tables `amis`.

Notez que les commandes MySQL, comme le langage SQL en général, sont très proches de l'anglais parlé : `show fields from amis;` veut dire "afficher les champs de amis". Il ne fait aucun doute que vous apprendrez rapidement à parler SQL couramment :)

Procédons par étape et entrons tout d'abord des informations dans la table `civil` :

```
mysql> INSERT INTO civil VALUES
```

```
mysql> show fields from amis;
+---------+----------------------+------+-----+---------+----------------+
| Field   | Type                 | Null | Key | Default | Extra          |
+---------+----------------------+------+-----+---------+----------------+
| acode   | smallint(5) unsigned |      | PRI | 0       | auto_increment |
| aciv    | tinyint(4)           | YES  |     | NULL    |                |
| anom    | varchar(80)          | YES  |     | NULL    |                |
| aprenom | varchar(80)          | YES  |     | NULL    |                |
| aadd1   | varchar(255)         | YES  |     | NULL    |                |
| aadd2   | varchar(255)         | YES  |     | NULL    |                |
| acp     | varchar(8)           | YES  |     | NULL    |                |
| aville  | varchar(80)          | YES  |     | NULL    |                |
| apays   | varchar(80)          | YES  |     | NULL    |                |
| atel    | varchar(20)          | YES  |     | NULL    |                |
| adate   | date                 | YES  |     | NULL    |                |
+---------+----------------------+------+-----+---------+----------------+
11 rows in set (0.07 sec)
```

```
('','Monsieur','Mr.');
Query OK, 1 row affected (0.11 sec)
```

Nous utilisons la clause *INSERT* pour insérer des informations. Encore une fois, c'est presque de l'anglais courant "insérer dans civil les valeurs (....)". Les valeurs insérées sont données dans leur ordre d'apparition dans la table. Nous omettons délibérément de donner une valeur pour le premier champ, ce qui provoque l'incrémentation pour la première insertion en utilisant le 0 par défaut.

Si nous répétons la requête pour une autre civilité, le premier champ sera incrémenté automatiquement :

```
mysql> INSERT INTO civil VALUES
('','Madame','Mme.');
Query OK, 1 row affected (0.00 sec)
```

Pour finir, ajoutons une dernière civilité :

```
mysql> INSERT INTO civil VALUES
('','Mademoiselle','Mlle.');
Query OK, 1 row affected (0.00 sec)
```

Nous pouvons ensuite lister le contenu de la table et constater que tout est correct (nous reviendrons au prochain chapitre sur les requêtes d'interrogation de la base) :

```
mysql> SELECT * from civil;
+-------+--------------+--------+
| ccode | cciv         | cabreg |
+-------+--------------+--------+
|     1 | Monsieur     | Mr.    |
|     2 | Madame       | Mme.   |
|     3 | Mademoiselle | Mlle.  |
+-------+--------------+--------+
3 rows in set (0.00 sec)
```

Nous avons bien nos trois enregistrements. Bravo, vous venez de remplir votre première table MySQL ! Continuons sur notre lancée avec des insertions dans la table `amis` :

```
mysql> INSERT INTO amis VALUES
('',1,'Dupond','Jean','12 rue de la
gare','','68000','COLMAR','FRANCE','','1974-06-
19');
Query OK, 1 row affected (0.00 sec)
```

En dehors du champ `acode`, qui est en auto-incrémentation, et de la date qui respecte le format imposé, nous remarquons des champs vides qui sont alors spécifiés par ''. La seule valeur numérique de l'insertion est le code de civilité qui se passe alors de '. Il existe une autre manière d'insérer des enregistrements dans une table. Il s'agit de préciser tout simplement les champs à utiliser :

```
mysql> INSERT INTO amis (aciv, anom, aprenom,
aadd1, acp, aville, apays, adate) VALUES
(2,'Durant','Marie','1 place de
garde','75016','PARIS','FRANCE','1978-12-05');
Query OK, 1 row affected (0.00 sec)
```

Dans cette requête d'insertion, nous spécifions, après le nom de la table, les champs concernés entre parenthèses. Les valeurs spécifiées ensuite par *VALUES* ne concernent que ces champs dans l'ordre utilisé précédemment. Les champs non utilisés sont laissés vides, à moins qu'une valeur par défaut n'ait été spécifiée dans la définition de la table. Le champ `acode` s'auto-incrémente sans problème.

Ajoutez dans votre table plusieurs autres enregistrements pour bien assimiler la syntaxe de la clause *INSERT* et pour pouvoir passer au chapitre suivant

```
mysql> SELECT * FROM amis;
+-------+------+----------+----------+--------------------+-------+-------+-----------+--------+-----------+------------+
| acode | aciv | anom     | aprenom  | aadd1              | aadd2 | acp   | aville    | apays  | atel      | adate      |
+-------+------+----------+----------+--------------------+-------+-------+-----------+--------+-----------+------------+
|     1 |    1 | Dupond   | Jean     | 12 rue de la gare  | NULL  | 68000 | COLMAR    | FRANCE | NULL      | 1974-06-19 |
|     2 |    2 | Durant   | Marie    | 1 place de garde   | NULL  | 75016 | PARIS     | FRANCE | NULL      | 1978-12-05 |
|     3 |    1 | Bodor    | Denis    | 6 rue de la scheer | NULL  | 67603 | SELESTAT  | FRANCE | 0388580208 | 1974-06-19 |
|     4 |    3 | Delatour | Michelle | 5 avenue de paris  | NULL  | 67000 | STRASBOURG | FRANCE | NULL      | 1978-09-29 |
|     5 |    1 | Duron    | Amédé    | 1 rue de la silice | NULL  | 8020  | TRUCMUCH  | SUISSE | NULL      | 1918-11-20 |
|     6 |    2 | Damett   | Cécile   | rue de la corneille | NULL | 18140 | TRIFOUILLY | FRANCE | NULL      | 1968-05-02 |
+-------+------+----------+----------+--------------------+-------+-------+-----------+--------+-----------+------------+
6 rows in set (0.00 sec)
```

concernant l'interrogation des tables. Les caractères accentués ne devraient pas vous poser de problèmes normalement. Il en va tout autrement des apostrophes. En effet, le caractère ' est utilisé pour marquer le début et la fin des chaînes de caractères. Si vous voulez utiliser ce caractère dans l'un des champs, il vous faudra l'échapper. Exemple : *'ruede l\'eau'*.

Créez plusieurs enregistrements comportant des champs vides et de préférence possédant des différences afin que nous puissions correctement poursuivre.

Requêtes d'interrogation

Les requêtes permettant de lister le contenu d'une table utilisent la clause *SELECT* (sélection). Une première requête de ce type et sans doute la plus simple est :
Voir tableau précédente.

Une fois n'est pas coutume, `select * from amis;` se traduit de l'anglais par "sélectionner tous les champs (*) depuis la table amis". La requête est des plus aisée mais parfaitement inutilisable pour une table contenant plusieurs dizaines ou centaines d'enregistrements. Voilà pourquoi il est possible d'ajouter une clause *WHERE* permettant de ne sélectionner qu'une partie des enregistrements en fonction de critères choisis.

De plus, une sélection des champs à afficher est possible. Ici, nous avons spécifié tous les champs avec *, mais nous pouvons les nommer en les séparant d'une virgule :

```
mysql> SELECT anom, aprenom, adate FROM amis;
+----------+----------+------------+
| anom     | aprenom  | adate      |
+----------+----------+------------+
| Dupond   | Jean     | 1974-06-19 |
| Durant   | Marie    | 1978-12-05 |
| Bodor    | Denis    | 1974-06-19 |
| Delatour | Michelle | 1978-09-29 |
| Duron    | Amédé    | 1918-11-20 |
| Damett   | Cécile   | 1968-05-02 |
+----------+----------+------------+
6 rows in set (0.00 sec)
```

Les enregistrements apparaissent avec les champs dans l'ordre spécifié dans la requête. Tentons maintenant une première sélection de l'enregistrement en demandant les noms, les prénoms et le pays des enregistrements dont le pays est la France :

```
mysql> SELECT anom, aprenom, apays FROM amis WHERE
apays='FRANCE';
+----------+----------+--------+
| anom     | aprenom  | apays  |
+----------+----------+--------+
| Dupond   | Jean     | FRANCE |
| Durant   | Marie    | FRANCE |
| Bodor    | Denis    | FRANCE |
| Delatour | Michelle | FRANCE |
| Damett   | Cécile   | FRANCE |
+----------+----------+--------+
5 rows in set (0.01 sec)
```

Notre requête traduite nous donne "sélectionner les champs anom, aprenom et apays depuis amis où apays est FRANCE". Nous constatons, en effet que Amédé Duron ne figure pas dans la liste. L'égalité n'est pas la seule possibilité dans notre besace, nous pouvons par exemple demander les nom et prénom de nos amis dont la date de naissance se situe après le 1er janvier 1977 :

```
mysql> SELECT anom, aprenom FROM amis WHERE
adate>'1977-01-01';
+----------+----------+
| anom     | aprenom  |
+----------+----------+
| Durant   | Marie    |
| Delatour | Michelle |
+----------+----------+
2 rows in set (0.00 sec)
```

Nous pouvons spécifier plusieurs critères de sélection dans la clause WHERE à l'aide d'opérateurs logiques :

```
mysql> SELECT anom, aprenom FROM amis WHERE
adate>'1920-01-01' AND adate<'1975-12-31';
+--------+---------+
| anom   | aprenom |
+--------+---------+
| Dupond | Jean    |
| Bodor  | Denis   |
| Damett | Cécile  |
+--------+---------+
3 rows in set (0.01 sec)
```

Ici, nous avons demandé les nom et prénom de nos amis dont la date de naissance est après le 1er janvier 1920 ET avant le 31 décembre 1975.

Essayez plusieurs imbrications de clauses WHERE avec des opérateurs logiques ET (AND) et OU (OR) jusqu'à devenir à l'aise avec cette clause.

Une dernière clause peut être utilisée ici et permet de spécifier un ordre d'affichage, c'est *ORDER BY* :

```
mysql> SELECT aprenom, adate FROM amis ORDER BY
adate;
+----------+------------+
| aprenom  | adate      |
+----------+------------+
| Amédé    | 1918-11-20 |
| Cécile   | 1968-05-02 |
| Jean     | 1974-06-19 |
| Denis    | 1974-06-19 |
| Michelle | 1978-09-29 |
| Marie    | 1978-12-05 |
+----------+------------+
6 rows in set (0.07 sec)
```

Le résultat est trié par ordre croissant. Pour inverser le tri, ajouter DESC (descendant, décroissant) en fin de requête :

```
mysql> SELECT aprenom, adate FROM amis ORDER BY
adate DESC;
```

Jointures

Une jointure est un lien reliant deux tables ou plus. Ce lien n'a pas d'existence propre au sein de la base de données. Ne vous attendez donc pas à tracer un trait reliant les tables ou à définir une jointure avec une commande spécifique. La jointure est matérialisée lors d'une requête et repose sur une liaison logique entre deux informations.

Nous avons ce cas en présence avec nos tables `civil` et `amis`. En effet, dans la table `amis`, nous faisons référence à un code permettant d'accéder aux informations de la table `civil`. La jointure est utile si, par exemple, nous voulons afficher la civilité, le nom et le prénom de nos amis :

```
mysql> SELECT cciv, anom, aprenom FROM civil,amis
WHERE ccode=aciv;
+--------------+----------+----------+
| cciv         | anom     | aprenom  |
+--------------+----------+----------+
| Monsieur     | Dupond   | Jean     |
| Madame       | Durant   | Marie    |
| Monsieur     | Bodor    | Denis    |
| Mademoiselle | Delatour | Michelle |
| Monsieur     | Duron    | Amédé    |
| Madame       | Damett   | Cécile   |
+--------------+----------+----------+
6 rows in set (0.00 sec)
```

Traduction de la requête : "Nous sélectionnons la civilité, le nom et le prénom depuis les tables civil et amis où le code de civilité de la table civil est le même que dans la table amis". La jointure est matérialisée par la clause WHERE qui établit la liaison entre `civil` et `amis`.

Le résultat est ainsi composé d'informations provenant de la table `civil` (`cciv`) et de la table `amis` (`anom`, `aprenom`).

Dans le cas où des champs porteraient le même nom, il est nécessaire de les préfixer par le nom de la table. Ce qui nous donnerait ici :

```
mysql> SELECT civil.cciv, amis.anom, amis.aprenom
FROM civil,amis WHERE civil.ccode=amis.aciv;
```

Autres clauses

Nous avons couvert jusqu'à présent le strict minimum des clauses disponibles. Mais pour que cette simple introduction à MySQL et à SQL soit complète, il nous faut ajouter quelques clauses supplémentaires à notre arc :

• La clause *DELETE* permet de supprimer des enregistrements. Attention, pensez à préciser un critère de sélection avec *WHERE* ; dans le cas contraire, vous supprimerez **TOUS LES ENREGISTREMENTS** de la table !

```
mysql> DELETE FROM amis WHERE anom='bodor';
Query OK, 1 row affected (0.00 sec)
```

• La clause *UPDATE* permet de modifier les valeurs des champs spécifiés. Là encore, utilisez *WHERE* pour sélectionner les enregistrements à modifier :

```
mysql> UPDATE amis SET aville='TRUCCHOSE' WHERE
anom='duron';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

• Les clauses *MAX*, *MIN*, *AVG* et *COUNT* permettent respectivement d'obtenir la valeur maximum ou minimum pour un champ, la moyenne et le nombre d'enregistrements dans une clause *SELECT* :

```
mysql> SELECT MAX(adate) FROM amis;
```

```
+------------+
| MAX(adate) |
+------------+
| 1978-12-05 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT MIN(adate) FROM amis;
+------------+
| MIN(adate) |
+------------+
| 1918-11-20 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT AVG(aciv) FROM amis;
+-----------+
| AVG(aciv) |
+-----------+
|    1.8000 |
+-----------+
1 row in set (0.00 sec)
```

L'exemple est mal choisi ici, car la moyenne des codes de civilité n'apporte aucune information utile.

```
mysql> SELECT COUNT(adate) FROM amis;
+--------------+
| COUNT(adate) |
+--------------+
|            5 |
+--------------+
1 row in set (0.00 sec)
```

Vous pouvez ajouter *DISTINCT* devant *COUNT* afin de ne compter que les enregistrements dont le champ est unique.

• La clause *GROUP BY* permet de grouper les enregistrements pour les clauses que nous venons de voir :

```
mysql> SELECT COUNT(apays), apays FROM amis GROUP
BY apays;
+--------------+--------+
| COUNT(apays) | apays  |
+--------------+--------+
|            4 | FRANCE |
|            1 | SUISSE |
+--------------+--------+
2 rows in set (0.00 sec)
```

Nous avons 4 amis en France et 1 en Suisse.

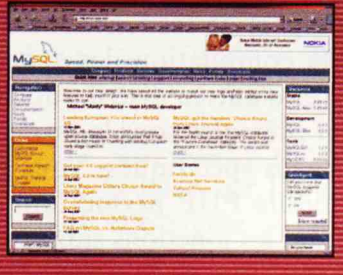
```
mysql> SELECT COUNT(cciv), cciv FROM civil,amis
WHERE ccode=aciv GROUP BY cciv;
+-------------+--------------+
| COUNT(cciv) | cciv         |
+-------------+--------------+
|           2 | Madame       |
|           1 | Mademoiselle |
|           2 | Monsieur     |
+-------------+--------------+
3 rows in set (0.00 sec)
```

Ici, nous mélangeons jointure et groupage pour obtenir le nombre d'enregistrements dans `amis` en fonction de leur civilité.

L'apprentissage du SQL est une tâche longue mais aisée. Nous n'avons présenté qu'une partie du langage ici, et je vous invite à consulter les différentes documentations existantes sur Internet et sous forme d'ouvrage spécialisé. Je suis persuadé que vous apprécierez les finesses du SQL et que vous prendrez plaisir à l'utiliser...

Liens

Mysql
http://www.mysql.com



Documentation complète de MySQL en français :
http://tecfa.unige.ch/guides/mysql/fr-man/manuel_toc.html

Des diapos de présentation d'un cours sur les bases de données :
http://www.iup.univ-avignon.fr/~Cours/SGBD/

PHP et MySQL : le duo de choc !

Si vous êtes arrivé jusqu'ici dans la lecture de ce hors série, c'est soit que vous venez de faire vos premiers pas avec Apache, PHP et MySQL, soit que vous avez déjà une connaissance de un ou plusieurs de ces éléments. Il est temps à présent de réunir toutes ces informations et de marier HTTP, HTML, PHP et MySQL pour construire votre site. Nous réutiliserons bon nombre d'éléments des précédents articles. N'hésitez donc pas à revenir en arrière si un point vous paraît obscur.

Pour les différents exemples qui vont suivre, nous allons réutiliser notre base d'amis définie dans l'article sur MySQL. Si vous connaissez déjà MySQL et son maniement, il conviendra d'adapter les identités (utilisateur, nom de la base, mot de passe) à votre installation. En ce qui concerne les tables en présence, la meilleure chose est encore de reprendre les structures que nous avons définies.

Installation

Le langage PHP dispose de fonctionnalités permettant son interfaçage avec les principaux SGBD disponibles. Le support de ces fonctionnalités est activable lors de la compilation des sources de PHP. Avec Debian (et d'autres distributions), ces fonctionnalités sont présentes sous la forme de modules dans des packages spécifiques. Vous devrez donc installer le module PHP permettant d'accéder à MySQL et offrant les fonctions supplémentaires :

```
# apt-get install php4-mysql
```

Le script de configuration lancé automatiquement après l'installation du package vous informe d'une manipulation indispensable au support MySQL pour PHP :

```
You are installing MySQL support for php4, but
it's not enabled in you /etc/php4/apache/php.ini
To enable it you need to add this line:

extension=mysql.so

Do you want me to add it now [y/N] ?
```

Répondez "y" à la question pour configurer le support dans le fichier de configuration de PHP (par défaut la réponse est non). Vous devrez ensuite relancer le service Apache pour prendre en compte ces modifications :

```
# /etc/init.d/apache restart
Reloading apache modules.
/usr/sbin/apachectl stop: httpd stopped
/usr/sbin/apachectl start: httpd started
```

Vous voilà prêt à utiliser MySQL depuis vos scripts PHP.

La base et l'ordre

Les fonctions MySQL de PHP vous permettront de vous connecter à un serveur MySQL en fonction à l'aide d'un identifiant et d'un mot de passe (si l'identifiant le demande). La fonction de connexion retourne un identifiant de connexion :

```
$dbh=mysql_connect("localhost", "wwwdata",
"mot_de_passe");
```

La fonction `mysql_connect` prend en argument trois paramètres : l'hôte où réside le serveur MySQL, un identifiant d'utilisateur MySQL valide et le mot de passe correspondant. Dès lors, la variable `$dbh` représente le lien avec le serveur MySQL. Peu importe ce que `$dbh` contient réellement, vous n'aurez jamais à modifier ce contenu. Vous utiliserez simplement `$dbh` comme point de connexion pour toutes les communications que vous aurez à établir avec le serveur MySQL.

Un site Web est rarement composé d'un seul et même fichier. Lorsque vous utilisez MySQL pour développer votre ou vos site(s), il est fort probable que plusieurs scripts PHP fassent appel à la fonction `mysql_connect`. Un bon programmeur est

feignant et s'il peut se passer de répéter plusieurs fois les mêmes paramètres, ce ne sera que bénéfique. En effet, imaginons que le mot de passe de l'utilisateur wwwdate change. En utilisant la ligne précédente telle quelle dans plusieurs fichiers PHP, vous serez obligé de modifier autant de fichiers qu'il y en a qui font appel à cette fonction.

La bonne technique est de placer ces paramètres de connexion dans un fichier à part (BASE.php par exemple). Celui-ci contiendra alors :

```
<?
$DBdatabase='mabase';
$DBuser='wwwdata';
$DBpass='mot_de_passe';
?>
```

Nous définissons trois variables contenant les paramètres qui risquent de changer : le nom de la base, le nom d'utilisateur MySQL et son mot de passe. Il ne nous reste plus, ensuite, qu'à utiliser la ligne suivante dans tous les fichiers PHP utilisant une connexion sur serveur MySQL :

```
require("BASE.php");
```

require vous permet de charger le contenu d'un autre fichier PHP dans le fichier courant (un peu comme le #include en C). Cette ligne ajoutée en tout début de fichier, les variables $DBdatabase, $DBuser et $DBpass seront utilisables en lieu et place des paramètres qu'elles remplacent :

```
$dbh=mysql_connect("localhost", "$DBuser",
"$DBpass");
```

Note : Nous n'avons pas pris la peine de stocker l'hôte MySQL dans une variable car, dans le cadre du présent magazine, c'est la même machine qui héberge le serveur HTTP et le serveur MySQL. Rien ne vous empêche cependant de placer l'hôte dans BASE.php et de vous en servir de la même manière. Notons également que require peut être très utile pour un site Web n'utilisant pas MySQL et très peu de PHP.

En effet, il arrive souvent que des morceaux entiers de code HTML soient récurrents dans les fichiers d'un serveur. Vous pouvez ainsi vous servir de require pour centraliser ces lignes HTML dans un fichier unique.

Première requête

Notre premier fichier aura pour but d'afficher le contenu de notre table amis de la base mabase. Nous allons tout d'abord nous connecter :

```
<html>
<body>
<?

require("BASE.php");

$dbh=mysql_connect("localhost", "$DBuser",
"$DBpass");
if (!$dbh) {
  echo "<font color=\"#FF0000\">ERREUR !
Impossible de se connecter à
$DBdatabase.</font><br>";
  echo "</body></html>");
  exit;
}
```

Nous prévoyons une éventuelle erreur et si $dbh reste indéfinie, nous affichons un message d'avertissement puis cessons l'interprétation du code. Si tout se passe bien, nous obtenons notre point d'accès à la base et pouvons continuer en lançant une requête :

```
$res=mysql_db_query("$DBdatabase", "SELECT anom,
aprenom, aville FROM amis;", $dbh);
```

La fonction mysql_db_query prend trois arguments : le nom de la base sur laquelle porte la requête, la requête elle-même et l'identifiant de connexion. La fonction retourne dans $res un identifiant ou objet de résultat. Cette variable n'est pas utilisable directement. N'essayez donc pas de l'afficher. Là encore, nous pouvons tester la présence d'une éventuelle erreur :

```
$errno=mysql_errno($dbh);
if ($errno!=0) {
  $erreur=mysql_error($dbh);
  echo "<font color=\"#FF0000\">$erreur</font>";
  mysql_close($dbh);
  echo "</body></html>";
  exit;
}
```

Nous ne pouvons pas tester directement la variable $res ici. Nous utilisons alors la fonction mysql_errno en lui passant comme paramètre l'identifiant de connexion. Cette fonction retourne

un numéro d'erreur ou 0 si aucune erreur n'a été rencontrée. Il nous suffit alors de tester le contenu de `$errno` pour afficher un message d'erreur que nous obtenons avec la fonction `mysql_error`. Nous n'oublions pas de fermer la connexion au serveur, d'afficher les balises HTML adéquates et de cesser l'interprétation du code PHP.

Si nous faisons délibérément une erreur dans le nom de la table `amis`, vous voyons que notre test fonctionne parfaitement et le navigateur affiche en rouge :

```
Table 'mabase.mis' doesn't exist
```

Sauf erreur, nous avons obtenu le résultat de la requête dans `$res`. Un résultat de requête est considéré comme une table et nous devons utiliser des fonctions spécifiques pour le traiter. Dans un premier temps, commençons par se renseigner sur le nombre de lignes du résultat :

```
$nbrligne=mysql_num_rows($res);
```

La fonction `mysql_num_rows` retourne le nombre de lignes de l'identifiant de résultat. Ce nombre nous permettra de parcourir les données obtenues.

Nous pouvons maintenant nous lancer dans une boucle pour récupérer et afficher ligne par ligne le résultat :

```
echo "<table border=2 cellpadding=2
width=\"100%\">";

for($i=0;$i<$nbrligne;$i++) {
  $row=mysql_fetch_row($res);
  echo
"<tr><td>$row[0]</td><td>$row[1]</td><td>$row[2]</
td></tr>";
}

echo "</table>";
```

Le résultat obtenu dépendra des données dans la table mais devrait être proche de la **figure 1**. Notre boucle `for` va utiliser 6 fois la fonction `mysql_fetch_row` permettant d'extraire une ligne du résultat via l'identifiant `$res`. `mysql_fetch_row` retourne un tableau à une dimension où chaque cellule est un champ du résultat. Cette fonction positionne ensuite automatiquement un pointeur sur la ligne suivante du résultat. De cette manière, un prochain appel à `mysql_fetch_row` provoquera l'extraction de la ligne suivante.

Fig.1

Dupond	Jean	COLMAR
Durant	Marie	PARIS
Bodor	Denis	SELESTAT
Delatour	Michelle	STRASBOURG
Duron	Amédé	TRUCCHOSE
Damett	Cécile	TRIFOUILLY

Ne considérez pas ces lignes comme utilisables pour toutes vos requêtes. Nous connaissons d'avance ici le nombre de champs qui compose une ligne du résultat. Avec une requête du type SELECT * FROM, vous n'êtes pas censé connaître ce nombre de champs. Voyons comment nous pouvons améliorer ce code et lui permettre d'être plus souple.

Nous avons utilisé `mysql_num_rows` pour connaître le nombre de lignes du résultat de la requête. Il existe une fonction équivalente pour le nombre de champs dans chaque ligne de résultat :

```
$nbrchamp=mysql_num_fields($res);
```

Nous avons maintenant une information supplémentaire, mais ce n'est toujours pas suffisant. Nous ne connaissons pas le nom des champs. Qu'à cela ne tienne, nous pouvons récupérer ces noms avec :

```
$nomchamp=mysql_field_name($res, numero_du_champ);
```

Cette fonction prend en argument l'identifiant de résultat et le numéro du champ dans le résultat puis retourne le nom du champ en question.

Nous avons maintenant tout ce qu'il nous faut pour réécrire un code plus souple :

- Le début du code ne change pas si ce n'est que la requête ou les noms des champs ont été remplacés par *

```
$dbh=mysql_connect("localhost", "$DBuser",
"$DBpass");
$res=mysql_db_query("$DBdatabase", "SELECT * FROM
amis;", $dbh);
```

```
$errno=mysql_errno($dbh);
if ($errno!=0) {
  $erreur=mysql_error($dbh);
  echo "<font color=\"#FF0000\">$erreur</font>";
  mysql_close($dbh);
  echo "</body></html>";
  exit;
}
```

• Nous récupérons les nombres qu'il nous faut pour nos boucles.

```
$nbrligne=mysql_num_rows($res);
$nbrchamp=mysql_num_fields($res);
```

• Nous commençons le tableau HTML.

```
echo "<p><p><table border=2 cellpadding=2
width=\"100%\">";
```

• Nous utilisons `$nbrchamp` et `mysql_field_name` pour dessiner l'en-tête du tableau.

```
for($i=0;$i<$nbrchamp;$i++) {
  printf
("<th>%s</th>",mysql_field_name($res,$i));
}
```

• Nous traitons ensuite ligne par ligne le résultat.

```
for($i=0;$i<$nbrligne;$i++) {
  $row=mysql_fetch_row($res);
```

• Et nous utilisons `foreach` pour parcourir le tableau retourné par `mysql_fetch_row`.

```
  echo "<tr>";
  foreach($row as $col) {
    if(!$col) $col=" ";
    echo "<td>$col</td>";
  }
  echo "</tr>";
}
```

Notez la condition `if` qui nous permet de conserver un aspect acceptable pour le tableau. En effet, un champ vide aura pour conséquence d'afficher une chaîne <td></td> qui n'est pas très esthétique dans le rendu du tableau. Dans le cas où un champ est vide, nous plaçons dans `$col` (contenu de la cellule courante) la chaîne `" "` (espace insécable).

• Enfin, nous terminons le tableau.

```
echo "</table>";
```

Le résultat est probant (**figure 2**) et nous pouvons l'appliquer à n'importe quelle table en changeant simplement la requête SQL.

Fig.2

acode	aciv	anom	aprenom	aadd1	aadd2	acp	aville	apays	atel	adate
1	1	Dupond	Jean	12 rue de la gare		68000	COLMAR	FRANCE		1974-06-19
2	2	Durant	Marie	1 place de garde		75016	PARIS	FRANCE		1978-12-05
7	1	Bodor	Denis	6 rue de la scheer		67603	SELESTAT	FRANCE	0388580208	1974-06-19
4	3	Delatour	Michelle	5 avenue de paris		67000	STRASBOURG	FRANCE		1978-09-29
5	1	Duron	Amédé	1 rue de la silice		8020	TRUCCHOSE	SUISSE		1918-11-20
6	2	Damett	Cécile	rue de la corneille		18140	TRIFOUILLY	FRANCE		1968-05-02

Utilisation avancée

Nous allons maintenant aborder deux concepts nouveaux en ce qui concerne HTML et PHP. Ces deux concepts sont fortement liés. Le premier concerne les formulaires HTML et le second la manière de traiter les informations provenant de ces formulaires avec PHP. Nous avons choisi de traiter les formulaires ici car ils ne seraient pas vraiment à leur place dans l'article sur la découverte du langage PHP. Par ailleurs, il n'y aurait aucune raison d'en parler dans l'introduction à HTML puisqu'ils n'ont pas vraiment d'intérêt dans le traitement ultérieur par un script PHP ou un CGI. C'est donc ici, dans l'article le plus orienté "pratique", que nous en parlons...

Un formulaire est un ensemble de balises permettant aux clients d'entrer des informations à destination du serveur. Il existe plusieurs balises différentes permettant d'entrer ces informations en fonction de la nature des informations et de la manière de les entrer.

Le début et la fin d'un formulaire sont marqués par les balises <FORM> et </FORM>. C'est à l'intérieur de la portée de cette balise que nous définirons les informations que l'utilisateur doit entrer. Les informations que l'utilisateur fournira seront envoyées via une méthode HTTP. Deux méthodes sont disponibles :

• GET : le contenu du formulaire est ajouté à l'URL ;

• POST : le contenu du formulaire est envoyé à part dans le flux HTTP.

Pour ce qui est de PHP, cela ne fait pas une grande différence quant à la manière de récupérer le contenu du formulaire. Le type de méthode à utiliser est spécifié par l'option `METHOD` dans la balise <FORM>. Une autre option permet de choisir une action à accomplir en envoyant le contenu du formulaire. Dans le cas d'un script PHP, l'action est tout simplement le fichier PHP capable de traiter les informations du formulaire. Ce qui nous donne, par exemple :

```
<FORM ACTION="formzou.php" METHOD="post">
```

Ici, notre fichier PHP est `formzou.php`. Je vous conseille de nommer de la même manière le formulaire et le fichier PHP traitant les informations qui en proviennent avec un suffixe (ici "zou").

Nous pouvons maintenant nous pencher sur les balises. Voyons tout d'abord une première balise permettant d'entrer une chaîne de caractères :

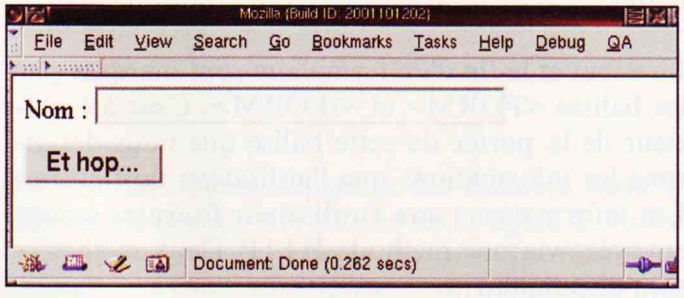
```
Nom : <INPUT TYPE="text" NAME="nom" SIZE=30><p>
```

Nous définissons ici une entrée (par défaut, il s'agit du type `text`) et une taille (ici 30 caractères). Pour provoquer l'envoi du formulaire, il nous faut ajouter un bouton d'envoi :

```
<INPUT TYPE="submit" VALUE="Et hop...">
```

Le type est alors un bouton de soumission de formulaire (`submit`) et sa valeur (`VALUE`) détermine le texte présent sur le bouton. Sur la fenêtre du navigateur, nous obtenons la **figure 3**.

Fig.3



Le code PHP contenu dans le fichier `formzou.php` n'a pas besoin d'utiliser d'instruction particulière. En effet, les noms des entrées du formulaire HTML (notés à l'aide de l'option NAME) deviennent automatiquement des variables dans le script PHP. Ainsi, si le client du site entre un texte dans notre champ et envoie le formulaire, une variable `$nom` existera automatiquement dans `formzou.php` et sera initialisée avec le contenu du champ correspondant :

```
<html>
<body>
<?
echo "<b>$nom</b><br>";
?>
</body>
</html>
```

Il nous suffira de l'afficher. Bien sûr, les balises `INPUT` de type `text` ne sont pas les seules utilisables :

Fig.4

- `checkbox` (**figure 4**)

```
<INPUT TYPE="checkbox" NAME="choix1"
VALUE="1">Un<br>
<INPUT TYPE="checkbox" NAME="choix2"
VALUE="1">Deux<br>
<INPUT TYPE="checkbox" NAME="choix3" VALUE="1"
CHECKED>Trois<br>
```

Le type `checkbox` permet d'afficher des cases que l'utilisateur peut cocher. Avec les `checkbox`, l'utilisateur peut en cocher plusieurs. L'option `CHECKED`, qui ne prend aucun paramètre, permet de cocher une ou plusieurs case(s) par défaut. Voici le code nécessaire pour traiter les informations :

```
if($choix1) {
  quelque chose...
}
```

- `radio` (**figure 5**)

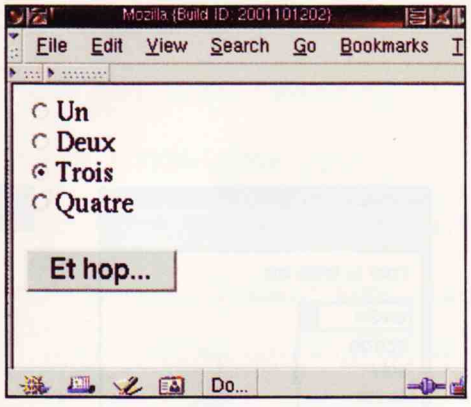


Fig.5

```
<INPUT TYPE="radio" NAME="choix" VALUE="1">Un<br>
<INPUT TYPE="radio" NAME="choix"
VALUE="2">Deux<br>
<INPUT TYPE="radio" NAME="choix" VALUE="3" CHEC-
KED>Trois<br>
<INPUT TYPE="radio" NAME="choix"
VALUE="4">Quatre<br>
```

Le type `radio` offre quasiment les mêmes possibilités que `checkbox` à la différence qu'une seule case pourra être cochée. Là encore, l'option `CHECKED` s'avèrera utile mais vous ne devrez l'utiliser qu'une seule fois. Le code PHP traitera l'information en utilisant la variable `$choix` automatiquement initialisé :

```
echo "$choix<br>";
```

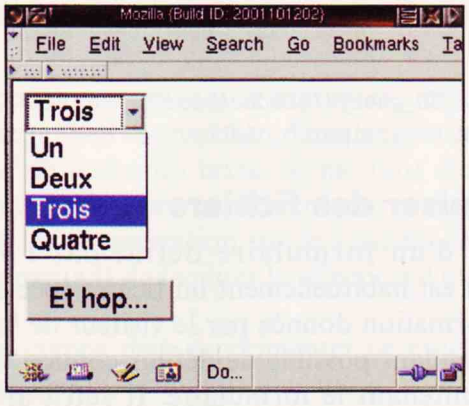


Fig.6

- <SELECT> (**figure 6**)

```
<SELECT NAME="choix">
  <OPTION VALUE="1">Un</OPTION>
  <OPTION VALUE="2">Deux</OPTION>
  <OPTION VALUE="3" SELECTED>Trois</OPTION>
  <OPTION VALUE="4">Quatre</OPTION>
</SELECT>
```

`SELECT` n'est pas un type mais une balise permettant d'offrir un choix sous la forme d'une liste déroulante. On définira dans la portée de `SELECT` un nombre défini de choix à proposer à l'aide de la balise `OPTION`. Ici `CHECKED` cède sa place à `SELECTED` afin de définir un élément de la liste par défaut.

Là encore, le code PHP est :

```
echo "$choix<br>";
```

Formulaires, PHP et MySQL

Vous pouvez bien sûr utiliser PHP et MySQL pour les scripts traitant les informations en provenance des formulaires HTML. Mais vous pouvez également utiliser la souplesse de PHP et l'efficacité de MySQL pour créer les formulaires. Commençons par un exemple simple ne faisant pas usage de MySQL :

```
<html>
<body>
<FORM ACTION="formzou.php" METHOD="get">

<?
echo "<SELECT NAME=\"choix\">";

for($i=1;$i<=7;$i++) {
  echo "<OPTION VALUE=\"$i\">$i</OPTION>";
}
echo "</SELECT>";
?>

<INPUT TYPE="submit" VALUE="Et hop...">
</FORM>
</body>
</html>
```

Fig.7

Ici, une simple boucle `for` nous permet de placer sept balises <OPTION></OPTION> dans le code HTML final envoyé au navigateur (**figure7**). Ce genre de code PHP est très intéressant pour les formulaires demandant au client de faire un choix numérique s'étalant sur une grande plage de valeurs.

De la même manière, l'utilisation des informations stockées dans une base MySQL est d'une grande utilité. En effet, un simple ajout, une modification ou encore une suppression dans une table suffit à changer un ou plusieurs formulaire(s) d'un site Web

et ce, sans modification d'une seule ligne de code HTML ou PHP.

Si nous prenons notre code PHP affichant tous les enregistrements d'une table, nous voyons que seule la requête change. Il nous suffit alors de créer un code générant un formulaire qui permet de créer les éléments de cette requête :

• Nous commençons par nous connecter à la base et lancer une requête sur la table :

```
<?
require("BASE.php");
$dbh=mysql_connect("localhost", "$DBuser",
"$DBpass");
$res=mysql_db_query("$DBdatabase", "SELECT * FROM
amis;", $dbh);
```

• Nous traitons une éventuelle erreur (on ne sait jamais) :

```
$errno=mysql_errno($dbh);
if ($errno!=0) {
  $erreur=mysql_error($dbh);
  echo "<font color=\"#FF0000\">$erreur</font>";
  mysql_close($dbh);
  echo "</body></html>";
  exit;
}
```

• Nous récupérons le nombre de champs :

```
$nbrchamp=mysql_num_fields($res);
```

• Nous débutons notre formulaire où nous récupérerons le choix dans une future variable `$tripar` :

```
echo "Trier la table sur :<p>";
echo "<FORM ACTION=\"base.php\" METHOD=\"get\">";
echo "<SELECT NAME=\"tripar\">";
```

• Nous définissons notre boucle proposant tous les noms de champs au visiteur de la page :

```
for($i=0;$i<$nbrchamp;$i++) {
  $courant=mysql_field_name($res,$i);
  echo "<OPTION
VALUE=\"$courant\">$courant</OPTION>";
}
```

• Et nous finissons le fichier :

```
echo "</SELECT>";
?>
<p><INPUT TYPE="submit" VALUE="Et hop...">
</FORM>
</body>
</html>
```

Fig.8

Dès le chargement de la page, une requête sera envoyée au serveur MySQL. Nous utilisons les informations retournées pour créer un formulaire (**figure 8**). Dans `base.php`, qui est notre dernier exemple permettant d'afficher le contenu d'une table, il nous suffira de modifier la ligne de requête de :

```
$res=mysql_db_query("$DBdatabase", "SELECT * FROM
amis;", $dbh);
```

en :

```
$res=mysql_db_query("$DBdatabase", "SELECT * FROM
amis ORDER BY $tripar;", $dbh);
```

Economiser des fichiers

La cible d'un formulaire défini par l'option `ACTION=` est habituellement un fichier spécial traitant l'information donnée par le visiteur de la page. Il est cependant possible de définir comme cible le fichier contenant le formulaire. Il suffit alors de faire un simple test sur une variable ne pouvant exister que si le formulaire a été envoyé.

On utilise alors un autre type d'entrée qui n'est pas visible sur le formulaire :

```
<INPUT TYPE="hidden" NAME="form" VALUE="1">
```

Il suffit ensuite de diviser le code en deux parties dépendantes de la variable `$form` :

```
<?
if($form) {
  echo "Vous avez entrez $nom<br>";
} else {
  echo "<FORM ACTION=\"form.php\"
METHOD=\"get\">";
  echo "<INPUT TYPE=\"text\" NAME=\"nom\"
SIZE=30><p>";
  echo "<INPUT TYPE=\"hidden\" NAME=\"form\"
VALUE=\"1\">";
  echo "<INPUT TYPE=\"submit\" VALUE=\"Et
hop...\">";
  echo "</FORM>";
}
?>
```

Un exemple concret plus avancé

Nous allons à présent cumuler toutes nos nouvelles connaissances pour écrire un code utilisant MySQL et une fonctionnalité avancée de PHP : nous allons créer une page permettant à un visiteur de fournir un fichier via le serveur HTTP. Les informations concernant ce fichier seront manipulées avec notre code PHP puis enregistrées dans une table de `mabase`.

Nous allons utiliser une option de la balise HTML <INPUT> dont nous n'avons pas encore parlé :

```
<INPUT TYPE="file" NAME="nomfich">
```

Cette option va provoquer automatiquement sur le navigateur Web l'apparition d'une entrée spécifique. Il s'agit d'un champ texte permettant d'entrer un chemin complet vers un fichier local et d'un bouton permettant la navigation sur le système de fichiers, ceci permettant de faciliter la sélection d'un fichier.

Nous pouvons déjà expérimenter ce nouveau type d'entrée avec le formulaire suivant :

```
<FROM METHOD="get" ACTION="imagezou.php" ENCTY-
PE="multipart/form-data">
<INPUT TYPE="file" NAME="fichier"><p>
<INPUT TYPE="submit" VALUE="Et hop...">
```

Pour ce qui est de l'entrée, il est impératif d'utiliser la méthode GET et de spécifier le type d'encodage utilisé (`multipart/form-data`). En l'absence de ces informations, le script PHP qui va suivre et qui traite l'information pourrait se comporter de manière aléatoire :

```
<?
echo "$fichier<br>$fichier_name<br>";
?>
```

Ici, nous ne faisons rien d'autre que d'afficher les informations utiles. Vous constaterez qu'en plus de la variable `$fichier` que nous sommes en droit d'attendre suite à la valeur de `NAME` dans le formulaire, nous obtenons une autre variable : `$fichier_name`.

Le fichier a été transmis correctement au serveur mais par souci de sécurité, celui-ci est automatiquement placé dans le répertoire temporaire du serveur et nommé de manière aléatoire.

La variable `$fichier` contient donc le nom du fichier tel qu'il est présent sur le serveur, avec le chemin absolu permettant de le trouver. Cette information est très importante puisqu'il s'agit de la seule manière de récupérer les données du fichier. De plus, la variable `$fichier_name` nous permet de connaître le nom original (sans chemin) du fichier.

Si vous essayez ce code, vous trouverez, par exemple, sur l'écran de votre navigateur quelque chose comme :

```
/tmp/php81DUaf
lm15e20.jpg
```

La première ligne est le nom du fichier sur le serveur et la seconde son nom sur le système de fichiers du client. En plus de la variable `$fichier_name`, une variable `$fichier_type` a également été créée. Elle contient le type de fichier tel qu'il a été donné par le navigateur client. Dans le cas présent (pour le fichier `lm15e20.jpg`), cette variable contiendrait `image/jpeg`.

Nous pourrions traiter le fichier de manière différente dans notre code en fonction de son type MIME. Le problème qui se pose alors est que nous devons entièrement faire confiance au navigateur client qui nous fournit cette information. Ainsi, avec Mozilla sous Linux, un fichier BMP sera décrit comme étant un type `application/octet-stream`, alors qu'il s'agit bien d'un format graphique.

L'étape suivante consistera à traiter les données. Imaginons que nous ayons sur notre serveur un

répertoire `fichiers` ; nous pouvons alors demander la copie du fichier du répertoire `tmp` vers `/var/www/fichiers` en lui redonnant son nom original :

```
<?
copy($fichier,
'/var/www/fichiers/'.$fichier_name);
echo "$fichier a été copié en
/home/www/fichiers/$fichier_name<br>";
?>
```

Nous obtenons sur la fenêtre du navigateur le message suivant :

```
/tmp/phpGZdGGq a été copié en
/home/www/fichiers/essai.jpg
```

Attention ! Le répertoire dans lequel vous allez copier vos fichiers à l'aide d'un code similaire ne doit pas être accessible sans mécanisme d'authentification. En effet, un client pourrait parfaitement connaître ou même supposer le répertoire où vous placez ces fichiers et vous envoyer un code PHP (comme `phpinfo()`). Ainsi, s'il accède à ce fichier ensuite, il va déclencher son interprétation et compromettre la sécurité de votre serveur ou de tout le système. Soyez vigilant !

Pour la partie concernant l'accès à `mabase`, nous allons rapidement créer une table avec `mysql` :

```
mysql> create table fichiers(
    -> code  TINYINT UNSIGNED DEFAULT '0' NOT NULL
auto_increment,
    -> utilisateur varchar(50),
    -> fichier varchar(150),
    -> PRIMARY KEY (code));
Query OK, 0 rows affected (0.00 sec)
```

Il ne nous reste plus, ensuite, qu'à faire l'insertion dans la table :

• On utilise le système habituel avec le fichier `BASE.php` contenant les informations sur le serveur MySQL :

```
<?
require("BASE.php");
```

- On copie le fichier reçu dans le répertoire `fichiers` avec son nom d'origine :

```
copy($fichier,
'/var/www/fichiers/'.$fichier_name);
```

• On se connecte au serveur MySQL et insérons un enregistrement (la variable `$REMOTE_ADDR` existe de fait, elle contient l'adresse IP de la machine cliente) :

```
$dbh=mysql_connect("localhost", "$DBuser",
"$DBpass");
$res=mysql_db_query("$DBdatabase", "INSERT INTO
fichiers VALUES
('','$REMOTE_ADDR','$fichier_name')", $dbh);
```

• On traite l'erreur :

```
$errno=mysql_errno($dbh);
if ($errno!=0) {
  $erreur=mysql_error($dbh);
  echo "<font color=\"#FF0000\">$erreur</font>";
  mysql_close($dbh);
  echo "</body></html>";
  exit;
}
```

• Nouveauté : nous récupérons dans le code PHP la valeur du champ `code` qui est auto-incrémenté avec la fonction `mysql_insert_id` :

```
$id=mysql_insert_id($dbh);
```

• Enfin, nous informons le visiteur de ce qui vient de se passer :

```
echo "Une entrée numéro $id a été ajoutée contenant un fichier $fichier_name provenant de $REMOTE_ADDR<br>";
?>
```

Vous venez d'avoir un aperçu pratique des possibilités offertes par le duo PHP et MySQL. Ceci est également applicable avec l'autre système de base de données libre PostgreSQL. Ce n'est qu'à l'usage que vous obtiendrez le meilleur de ce mélange et je vous invite à user et abuser des bases de données avec PHP.

D'expérience, vous en arriverez à ne plus pouvoir concevoir un site Web dans utiliser le SGBD tant ce genre de système est performant et souple.

Liens

PHP :
http://www.php.net

MySQL :
http://www.mysql.com

SSL : Secure Sockets Layer

Sécuriser la communication entre un serveur Web et un navigateur client est parfois utile, en particulier lorsqu'il s'agit d'informations sensibles (commerce électronique, intranet professionnel, etc.). Vous avez sans doute déjà visité des sites où le petit cadenas du navigateur était fermé, c'est précisément de cela qu'il s'agit : HTTPS, les transactions HTTP sécurisées via SSL.

SSL

SSL utilise une technique de chiffrement à clef publique. Pour sécuriser la connexion, le serveur envoie au client une clef publique que celui-ci utilisera pour chiffrer les données que seul le serveur pourra déchiffrer grâce à sa clef privée. L'étape suivante consiste éventuellement pour le client à envoyer de manière chiffrée sa clef publique (à la demande du serveur). Ainsi, serveur et client peuvent s'authentifier mutuellement et s'assurer de l'identité de chacun. Enfin, client et serveur négocient et partagent une clef et une méthode de chiffrement pour le transfert des données (clef de session).

HTTPS est le terme utilisé pour définir un protocole HTTP "over" SSL. Le flux de données HTTP qui transite entre le serveur et le client est chiffré. Mis à part la sécurité offerte par HTTPS, on notera une différence fondamentale avec HTTP : le port. En effet, alors que le protocole HTTP utilise le port 80, HTTPS utilise par défaut le port 443.

Il existe de nombreuses implémentations HTTPS. Les deux solutions "libres" sont mod_ssl et Apache-SSL. Le premier est une dérivation du second. Chaque solution possède ses avantages et ses inconvénients ; il n'est pas question ici de nous lancer dans le troll ;) On notera cependant que mod_ssl n'est pas le successeur d'Apache-SSL : il s'agit simplement de deux implémentations différentes s'utilisant quasiment de la même manière. Les tests et les manipulations qui figurent dans cet article utilisent mod_ssl mais peuvent être appliqués avec son alter ego.

Installation

Par défaut, le package Apache ne contient pas de support pour le chiffrement SSL. Il vous faudra donc installer un nouveau package :

```
# apt-get install apache-ssl
```

Cette commande aura pour effet d'installer un nouveau serveur Apache supportant SSL. Debian propose ainsi de faire fonctionner deux services Apache sur votre machine. Le premier étant un serveur HTTP classique et l'autre un serveur HTTPS. Vous trouverez les fichiers de configuration du nouveau serveur dans le répertoire `/etc/apache-ssl` et le script d'init sous le nom `/etc/init.d/apacha-ssl`. Les fichiers de données du nouveau serveur seront également placés dans `/var/www`.

Configuration

Lors de la procédure d'installation, le script de configuration vous pose plusieurs questions dont dépendra la configuration du serveur :

• Dans un premier temps, le script de configuration va générer une nouvelle paire de clefs :

```
creating selfsingned certificate
replace it with one signed by a certification authority (CA)
enter your ServerName at the Common Name prompt
If you want your certificate to expire after x days call this
programm
with -days x
Using configuration from /usr/share/doc/
apache-ssl/examples
/ssleay.cnf
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to '/etc/apache-ssl/apache.pem'
```

• Ensuite, une série de questions vous sera posée :

```
You are about to be asked to enter information that will be
incorporated
```

```
into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

• Vous pouvez laisser les champs vides, mais il est conseillé de les remplir. Tout d'abord, on vous demande le code à 2 chiffres du pays (FR), un état (mettez none pour la France), une ville, une société ou organisation et un secteur d'activité dans la société :

```
Country Name (2 letter code) [GB]:
State or Province Name (full name) [Some-State]:none
Locality Name (eg, city) []:selestat
Organization Name (eg, company; recommended) []:diamond
Organizational Unit Name (eg, section) []:redac
```

• Cette question est très importante. Afin de générer une paire de clefs valide pour votre serveur, la procédure de configuration a besoin du FQDN de votre serveur, c'est-à-dire son nom de domaine pleinement qualifié. Vous devez mettre ici le nom complet de la machine tel qu'il sera visible depuis Internet :

```
server name (eg. ssl.domain.tld; required!!!) []:egon
```

• Enfin, l'adresse *email* du responsable du serveur est demandée :

```
Email Address []:webmaster@egon
```

La procédure suit alors son cours :

```
Installing new configuration file /etc/apache-ssl/httpd.conf
...
Installing new configuration file /etc/apache-ssl/cron.conf ...
Correcting mime_module loading.
An Apache-SSL configuration exists, but needs some tweaking.
Your config files will not be modified until you select Y at
"save changes."
The ServerAdmin is set to webmaster@egon.
The DocumentRoot is set to /var/www.
Leaving existing site /var/www/index.html untouched.
Finding DSO
mods.....found.
```

Vous pouvez constater que les modules permettant le support SSL d'Apache sont effectivement configurés et utilisés :

```
[...]
LoadModule config_log_module
/usr/lib/apache/1.3/mod_log_config_ssl.so
LoadModule auth_module /usr/lib/apache/1.3/mod_auth_ssl.so
[...]
```

• Le système de configuration vous demande alors si vous voulez enregistrer la configuration :

```
Pondering..... done.
Save these changes to the configuration files? [Y/n]
```

Mise en œuvre

Nous considèrerons dans cette partie que vous avez une connaissance minimum du fichier de configuration d'Apache. Nous allons décrire les différentes options automatiquement définies par la procédure d'installation. En principe, si vous avez répondu correctement à toutes les questions précédentes, le serveur est directement utilisable.

Ajoutons d'abord toutes les directives pour que Apache puisse correctement fonctionner avec SSL :

• On définit tout d'abord le port TCP/IP à écouter pour les requêtes HTTP/SSL. Par convention, c'est le port 443 qui est utilisé :

```
Listen 443
```

• On spécifie ensuite le répertoire contenant le certificat SSL et le nom de fichier de certification lui-même :

```
SSLCACertificatePath /etc/apache-ssl
SSLCertificateFile /etc/apache-ssl/apache.pem
```

• Nous avons installé un certificat pour le serveur, mais il est également possible de demander une certification au client afin de s'assurer de son identité :

```
SSLVerifyClient 0
```

Cette option prend en argument une valeur définissant le type de vérification :

0	Aucune certification n'est nécessaire.
1	Le client peut présenter un certificat.
2	Le client DOIT présenter un certificat valide.
3	Le client peut présenter un certificat valide mais l'autorité de certification (CA) n'est pas nécessairement valide.

• Enfin, nous n'oublions pas d'activer le support SSL :

```
SSLEnable
```

Hôte virtuel

Pour l'heure, si vous avez installé les packages Apache et Apache-SSL, une requête arrivant sur le port 80 sera traitée par le service Apache et sur le port 443 par le service Apache-SSL. Deux serveurs HTTP sur une petite configuration CPU peuvent poser des problèmes de performance.

La solution consiste alors à supprimer le service Apache et à ne conserver que le serveur HTTP supportant SSL :

```
# apt-get remove apache
```

Dès lors, les connexions sur le port 80 ne seront plus traitées et nous devons réécrire la configuration du serveur. La technique utilisée est de laisser tourner le serveur HTTP sur le port 80 et de créer un hôte virtuel attendant les requêtes sur le port 443 :

```
<VirtualHost _default_:443>
DocumentRoot /var/www/secu
DirectoryIndex index.html
ErrorLog /var/log/apache/error.secu.log
TransferLog /var/log/apache/access_secu.log
```

Ces directives n'ont aucun rapport avec SSL ; nous n'avons fait qu'associer un hôte virtuel avec le port 443. Donc, les données sont stockées dans `/usr/local/httpd/secu` avec un fichier `index.html` comme page par défaut.

Nous ajoutons ensuite les directives permettant de supporter SSL pour cet hôte virtuel :

```
SSLEngine
SSLCACertificatePath /etc/apache-ssl
SSLCertificateFile /etc/apache-ssl/apache.pem

</VirtualHost>
```

Vous pouvez maintenant relancer le serveur Apache-SSL.

Ici, notre serveur est `egon`. En pointant notre navigateur sur **http://egon**, nous accédons au serveur via le port 80. En revanche, en utilisant **https://egon**, nous indiquons au navigateur d'utiliser le port SSL par défaut (443) et nous arrivons sur l'hôte virtuel.

Les certificats signés

Avec le certificat que nous avons généré via la procédure d'installation, seule une partie des fonctionnalités HTTPS est utilisée. En effet, le client n'a aucune preuve que notre certificat "auto-signé" possède une quelconque valeur. Netscape le signale de manière assez explicite : "However, Netscape does not recognize the authority who signed its Certificate".

Ceci n'est pas très gênant dans le cas d'un serveur Intranet ou Internet limité à un cercle de confiance restreint, car l'autorité se trouve simplement dans la même société, voire dans le même local. Dans le cas d'un serveur Internet "de production", ce genre de message peut semer le doute dans la tête de l'internaute.

Une autorité de certification est un tiers de confiance qui a pour rôle d'établir un lien entre une certification et une personne physique ou morale. Ce tiers est le garant de la validité des informations que fournit le serveur au client.

Vous ne pouvez pas faire valider un certificat que vous avez généré par une autorité de certification. Ces tiers de confiance délivrent des certificats signés aux serveurs HTTPS. La procédure est habituellement payante.

Il existe deux organismes faisant office de référence pour cela : Verisign et Thawte (prononcez "taute"). En fait, comme signalé sur la page d'accueil du site français de Thawte, il s'agit plus ou moins de la même entité : "Thawte, groupe VeriSign, n°1 français de la certification SSL". Les tarifications ont de quoi faire pâlir le particulier soucieux de proposer un minimum de sécurité sur son site : 125$ la première année (100$/an ensuite) pour l'offre de base, ce qui nous porte à presque 1000 francs avec le cours actuel de la devise.

Liens

Serveur Apache
http://www.apache.org
Apache-SSL
http://www.apache-ssl.org
mod_ssl
http://www.modssl.org
OpenSSL
http://www.openssl.org
Thawte
http://www.thawte.com.fr/
Verisign
http://www.verisign.com/server/

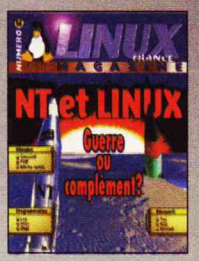
Complétez votre collection Linux Magazine et Linux Magazine Hors-Série

Linux Magazine N°8 :
Résolument Objet ! Découvrez le langage Eiffel, Corba, Gnome... Connectez un afficheur LCD sur votre port parallèle et continuez votre apprentissage des langages qui font la force de Linux.

Linux Magazine N°9 :
Linux est-il prêt pour le jeu ? Réponses des auteurs de Xracer, Hopkins FBI et ALE Clone. Découvrez un environnement Objet très puissant : GNUstep et apprenez à créer des plug-ins Netscape.

Linux Magazine N°10 :
Poursuivez l'exploration des différents langages comme Obj, Scheme, C et autres toolkits comme GTK+ et GNOME. La suite de la présentation de GNUstep ainsi que de nouveaux articles sur la sécurité réseau ou encore l'apprentissage de LaTex.

Linux Magazine N°11 :
Retrouvez vos rubriques habituelles de programmation (GNOME, Gtk, Scheme, C...). Faites connaissance dans ce numéro avec les applications GNUstep, les ports séries, PHP, et bien d'autres choses...



Linux Magazine N°12 :
Ce numéro vous propose un dossier complet consacré à l'analyse de la situation entre Linux et Windows NT. Également des interviews des représentants de Corel, IBM, Sun et toutes les rubriques habituelles.

Linux Magazine N°13 :
Nous vous proposons un dossier sur le cryptage grâce à SSF. Vous découvrirez les nouveautés de KDE 2.0 et vous pourrez continuer votre ballade à travers les langages de programmation avec C, Scheme, Lua, KDE, Gnome, Gtk+...

Linux Magazine N°14 :
Le futur est à notre porte et la communauté linuxienne n'hésitera pas à la franchir. Un numéro de Linux Magazine France qui parle de l'avenir. Mais aussi des choses qui font que Linux est un monde où l'on ne s'ennuie pas.

Linux Magazine N°15 :
Faites vos premiers pas dans le monde des clusters. Côté programmation, nous ferons une incursion dans le monde de l'audio, nous développerons un nouveau module PAM et toujours notre balade dans l'univers Scheme, du C, de GNOME, d'Imlib et de Gtk+.

Linux Magazine N°16 :
Nous verrons, dans ce numéro, comment bien composer son CV avec LaTeX. Une nouvelle section consacrée au Hurd fait son apparition dans la rubrique "Actu". Vous pourrez suivre Le développement et l'évolution de Hurd.

Linux Magazine N°17 :
Commencer votre apprentissage avec un langage de plus en plus populaire : Python. Nous continuerons notre chemin dans les méandres du C, de Scheme, de GTK, etc. Enfin, les amateurs de puissance apprécieront sans doute la seconde partie de l'article sur les clusters où il est davantage question de pratique.

Linux Magazine N°18 :
Nous découvrons dans ce numéro Nedit, un éditeur de texte plein de ressources, le système de fichiers/proc et scios. Nous continuons toujours l'exploration des clusters et nous ferons connaissance avec le moteur d'indexation et de recherche Ht://Dig.

Linux Magazine N°19 :
Pour cette parution d'été nous vous avons concoté un dossier sur PostgreSQ2, un Système de Gestion de Bases de Données Relationnelles libre et puissant. Vous découvrirez le langage SQL et la manière de créer des applications utilisant PostgreSQL.

Linux Magazine N°20 :
Le dossier de ce mois est consacré au langage PHP permettant de réaliser très simplement des sites Web réellement dynamiques.

Linux Magazine N°21 :
Vous découvrirez avec ce numéro tout ce qu'il faut savoir sur IPv6 et la manière de l'utiliser sous Linux. Vous apprendrez également comment sécuriser l'accès à une machine GNU/Linux, configurer une station Xwindow avec xdm, résoudre et optimiser avec SciLa ou encore pourquoi le GNU/Hurd est si différent...

Linux Magazine N°22 :
Dossier : GNU Portable Threads ; Les threads avec Perl. Rendu réaliste en dessin vectoriel, transparence réelle et transparence simulée ; Bash, soyez plus à l'aise dans votre coquille ; Construction d'un système Linux embarqué ; Créez votre OS avec OSKit ; Comment convertir Mandrake, RedHat en Debian

Linux Magazine N°23 :
Les outils pour débugger sous Linux. Eviter les failles de sécurité dès le développement d'une application ; Graphes et réseaux avec Scilab, la boite à outils Metanet, 1ere partie ; Unicode, tous les textes pour toutes les langues

Linux Magazine N°24 :
Développer des applications PalmOS ; POSE : l'émulateur de Palm ; Faire cohabiter Palm et Linux. Lancer une application dès le démarrage du système. Création et gestion simple de listes de diffusion. Postfix, une alternative à Sendmail ; Partager une connexion Internet via un port série.

Linux Magazine Hors-Série N°2 :
Corel WordPerfect 8
Voici le traitement de texte le plus utilisé à travers le monde maintenant disponible sous Linux. Découvrez dans ce numéro toute la procédure d'installation, plus de 90 fonctions de tableur intégrées, etc. Mais aussi ApplixWare 4.4.1 en version trial, Interbase 4.0 en version complète, Ingres 2 Beta en version complète Xfree 3.3.3.

Linux Magazine Hors-Série N°3 :
Spécial 3D et gravage :
Retrouvez les principaux logiciels concernant la 3D. Après une petite interview avec le concepteur de chaque logiciel, vous allez pouvoir découvrir les fantastiques possibilités que vous offre la 3D sous Linux.

Linux Magazine Hors-Série N°4 :
SuSE Linux 6.2
Un numéro spécial entièrement consacré à la dernière distribution de la société SuSE. Sur le CD accompagnant ce magazine, vous trouverez une version d'évaluation de la SuSE 6.2, la version 6.1 de DB2 d'IBM avec comme toujours une explication détaillée sur l'installation et l'utilisation de cette application.

Linux Magazine Hors-Série N°5 :
Slackware 7.0
Cette distribution n'est pas vraiment destinée aux novices, mais plutôt aux utilisateurs accomplis. En effet, avec la Slackware 7.0, pas d'installation automatisée, tout se fait à la main au profit de la stabilité du système, pour le plus grand plaisir des puristes !

Linux Magazine Spécial Debian :

Découvrez une distribution étonnante à plus d'un titre. La Debian 2.2 possède un système de mise à jour et d'installation de nouveaux logiciels unique et extrêmement performant. En effet celui-ci vous permet de garder votre système en permanence upgradé via internet, les cd-rom, ou tout autre média susceptible de contenir des package debian.

Offre spéciale 39 F (5,95€)

Linux Magazine Hors-Série 6 :

Comparez et forgez votre propre opinion concernant les nouvelles versions des deux leaders des distributions Linux. Ce match des 7.0 vous permettra de choisir laquelle de ces distributions vous convient le mieux. Vous trouverez dans les pages du magazine les procédures d'installation, des explications concernant la configuration du système et une critique complète de la Red Hat 7.0 et de la SuSE 7.0

Commandez aussi vos anciens numéros sur le site

www.ed-diamond.com

Bon de commande des anciens numéros

Magazine	Prix N°	Quantité	Total
Linux Mag N°3	20 F (3,05€)		
Linux Mag N°4	20 F (3,05€)		
Linux Mag N°5	20 F (3,05€)		
Linux Mag N°6	20 F (3,05€)		
Linux Mag N°7	20 F (3,05€)		
Linux magazine avec CD : A partir du n°8 Linux Mag est disponible uniquement avec CD			
Linux Mag N°8	35 F (5,34€)		
Linux Mag N° 9	35 F (5,34€)		
Linux Mag N°10	35 F (5,34€)		
Linux Mag N°11	35 F (5,34€)		
Linux Mag N°12	35 F (5,34€)		
Linux Mag N°13	35 F (5,34€)		
Linux Mag N°14	35 F (5,34€)		
Linux Mag N°15	35 F (5,34€)		
Linux Mag N°16	35 F (5,34€)		
Linux Mag N°17	35 F (5,34€)		
Linux Mag N°18	35 F (5,34€)		
Linux Mag N°19	35 F (5,34€)		
Linux Mag N°20	35 F (5,34€)		
Linux Mag N°21	35 F (5,34€)		
Linux Mag N°22	35 F (5,34€)		
Linux Mag N°23	35 F (5,34€)		
Linux Mag N°24	35 F (5,34€)		
Linux Mag N°25	35 F (5,34€)		
Linux Mag N°26	35 F (5,34€)		
Linux Mag N°27	35 F (5,34€)		
Linux Mag N°28	35 F (5,34€)		
Linux Mag N°29	35 F (5,34€)		
Linux Mag N°30	35 F (5,34€)		
Linux Mag N°31	35 F (5,34€)		
Linux Mag N°32	35 F (5,34€)		
Linux Magazine Hors-Série			
Linux Mag Hors-Série 2	35 F (5,34€)		
Linux Mag Hors-Série 3	35 F (5,34€)		
Linux Mag Hors-Série 4	35 F (5,34€)		
Linux Mag Hors-Série 5	35 F (5,34€)		
Linux Mag Spécial Débian	39 F (5,95€)		
Linux Mag Hors-Série 6	49 F (7,47€)		

Frais de port : France métropolitaine 25 F (3,81 €) U.E. plus Suisse, Liechtenchtein, Maroc, Tunisie, Algérie 35 F (5,34 €)

Total	
Frais de port	
Total de la commande	

Mode de règlement

Total Commande de magazines

☐ **Carte bancaire** — **Numéro :** _____/_____/_____/_____

Date d'expiration _____/_____

☐ **Chèque bancaire**

☐ **Chèque postal**

Signature :

Frais de port Dom-Tom et autres pays, nous contacter au 03 88 58 02 08

NOM ..... **PRÉNOM** .....

ADRESSE .....

.....

CODE POSTAL ..... **VILLE** .....

Linux Magazine France
est édité par Diamond Editions
B.P. 121 - 67603 Sélestat Cedex
Tél. : 03 88 58 02 08
Fax : 03 88 58 02 09
E-mail : lecteurs@linuxmag-france.org
service commercial : abo@linuxmag-france.org
Site : www.linuxmag-france.org

Directeur de publication : Arnaud Metzler
Rédaction
Rédacteur en chef : Denis Bodor
Secrétaires de rédaction : Carole Durocher
Conception graphique : Pascale Bauer
Impression : Didier Québécor - Strasbourg
Responsable publicité :
Jessie Quirin
Tél. : 03 88 58 02 08
Distribution :
(uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04
Service des ventes : Distri-médias :
Tél. : 05 61 72 76 24
Distribution Belgique :
Tondeur Diffusion
Avenue Van Kalken, 9
1070 Bruxelles
Press@tondeur.be
Service abonnement :
Tél. : 03 88 58 02 08
Conditionnement : Bensha
62, av de Belgique -68110 Illzach-Modenheim
Tél. : 03 89 61 77 17 - Fax. 03 89 61 77 19

Dans le respect de l'esprit des logiciels libres :

1- Le prix de vente du présent CD-Rom et magazine correspond uniquement aux frais d'impression de ces supports, de gestion des envois, de port, les logiciels étant mis gracieusement à la disposition des utilisateurs.

2- La mise en oeuvre et l'utilisation des logiciels et applicatifs figurant sur les CD-Rom distribués par Linux Magazine, est faite sous la pleine et entière responsabilité de l'utilisateur de ces logiciels. A ce titre, l'utilisation de ces logiciels et applicatifs mis à disposition par Linux Magazine implique, de la part des utilisateurs, l'acceptation tacite de la renonciation à tout recours à l'encontre de Linux Magazine et de ses éditeurs, quel que soit le préjudice subi par l'utilisateur.

Dépôt légal : 3e Trimestre 1998
N° ISSN : 0183-0864
Commission Paritaire : en cours
Périodicité : Bimestriel
Prix de vente : 35 FF.

PEARL

6, rue de la Scheer - ZI Nord - 67603 SELESTAT

www.pearl.fr

0,78 F TTC/mn
N°Indigo 0 820 822 823

GRATUIT !
Demandez notre catalogue 80 pages, par téléphone, fax, internet ou minitel !

Écran 17" numérique Goldstar SW775N

Ecran plat coins carrés 17" antireflet ○ Résolution jusqu'à 1280 x 1204 Pitch 0,27 mm ○ Fréquences verticales jusqu'à 160 Hz ○ Compatible VGA, SVGA, XGA, EVGA ○ Réglages complets à l'écran (OSD+30 mémoires) ○ Compatible MAC ○ Conforme à MPRII, NUTEK, CE et Energy Star

Réf.G10

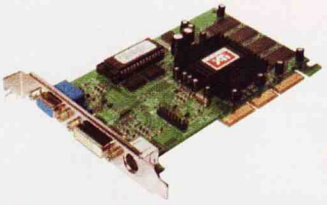
Compatible LINUX

1799,00 TTC
272,88 €

LG

ATI Radeon 64 Mo

Sortie TV ►Format AGP
Réf. PC682



Compatible LINUX

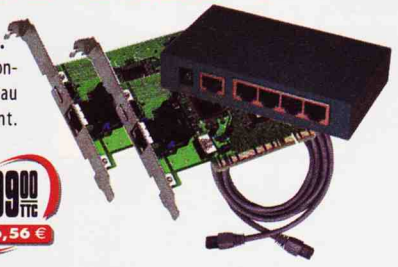
699,00 TTC
106,56 €

Kit réseau PCI 100-Base-T

Une solution réseau rapide, flexible et moderne pour deux postes. Ce kit évolutif est adapté aussi bien aux jeux qu'aux activités professionnelles. Grâce au Hub, il vous sera possible de connecter à votre réseau d'autres PC à tout moment sans en perturber le fonctionnement.

Comprend : ○2 cartes PCI 100 Base T ○ 2 câbles RJ45 ○1 Hub 100Base TX 4+1

Réf. PE225



Compatible LINUX

699,00 TTC
106,56 €

Nous vous proposons de nombreux autres produits pour LINUX !

Bon de Commande à retourner à PEARL Diffusion à l'adresse ci-dessous

Quantité	Désignation	Prix Unitaire	Prix Total

LMH59

6, rue de la Scheer - B.P. 121
ZI Nord - 67603 SELESTAT
Tél : 03 88 58 02 02
Fax : 03 88 58 02 07

TOTAL :
Frais de port (logiciels) : 39 F/ 5,95 €
Frais de port (matériel) : 49 F/ 7,47 €
Si contre remboursement : +45 F/ 6,68 €
Option Chronopost : +45 F/ 6,86 €
TOTAL A PAYER :

Nom & Prénom _____
Adresse _____
Code postal / Ville _____
Signature
Mode de paiement :
__Chèque (Uniquement par courrier) __Mandat __Contre remboursement
__Carte bancaire : N° : _____ / _____ / _____ / _____
Date d'expiration : _____ / _____

Commandes et devis administratifs uniquement par courrier - Pas de livraison dans les DOM TOM et Hors Europe

NAPA DAV310 : Lecteur CD/MP3/CDVidéo

Le DAV310, en plus de lire les CD audio et Mp3, vous permettra de visionner des vidéos CD. Vous pourrez ainsi préparer vos présentations sur votre PC , et sans avoir recours à une installation lourde vous pourrez les lire sur n'importe quel téléviseur.

Description : ▶ Lecteur CD/MP3/VCD portable
▶ Dimensions : 164x146x31mm
▶ Alimentation : batterie li-ion et adaptateur secteur inclus
▶ Fonction OSD
▶ Écran LCD de contrôle.

inclus :
▶ Écouteurs stéréo
▶ Télécommande
▶ Câble vidéo

121,81 € TTC — 799 F

Réf. PE931

Scanexpress 1200 UB

▶ Surface utile : 216 x 297 mm ▶ 36 bits (68 milliards de couleurs) ▶ 4096 Niveaux de gris ▶ Résolution : 600 x 1200 ppp optique et interpolée : 19200 x 19200 ppp ▶ Inclus Textbridge OCR et Picture Publisher **Réf. S5021**

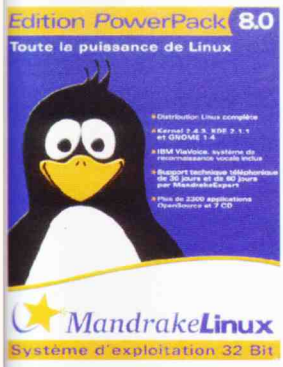
76,07 € TTC — 499 F

Clavier souple et étanche

Ne craignez plus les tasses de café qui tombent sur votre clavier. Grâce à ce tout nouveau modèle **transparent et étanche**, vous ne risquerez plus les dommages liés à des projections liquides, grasses,... **Idéal en milieu poussiéreux ou humide**. De plus, il se rangera n'importe où grâce à sa strucure souple. **Vous pourrez même le rouler** pour le ranger ou le transporter. **Ultra plat** (485x148x8mm), il est muni d'une prise PS2.

Réf. PE8963

45,58 € TTC — 299 F



LINUX Mandrake 8.1

Débutant

Cette distribution basée sur le kernel 2.4.3 comprend StarOffice 5.2, ViaVoice (version complète), The Gimp, Mozilla, Kmail, Konqueror, Acrobat Reader, Flash, Netscape, Java 2, des jeux et bien d'autres logiciels. Le support d'un grand nombre de cartes graphiques 3D et de l'USB en font un outil puissant et complet.

Linux Mandrake version standard **Réf. LI806 Prix : 33 €/216,47 F**

Linux Mandrake Powerpack Ce pack très complet inclus les meilleures applications Open Source et commerciales disponibles.
Réf. LI807 Prix : 69 €/452,61 F

Linux Mandrake Pro suite La solution Linux pour l'entreprise. Offrant un support technique étendu.
Réf.LI808 Prix : 169,95€/1114,80F

Baisse de Prix

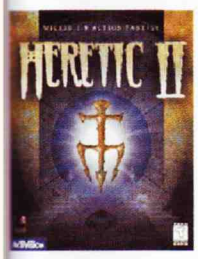
33,00 € TTC — 216,47 F

HERETIC II

Jeux

Evoluez dans un environnement hostile en 3D et utilisez vos pouvoirs magiques pour vaincre les demons qui ont pris possession du monde. Bienvenue dans votre nouvelle religion !

Réf. LI500



VALABLE JUSQU'A EPUISEMENT DE STOCK

30,34 € TTC — 199 F

MYTH II Soulblighter

Jeux

Vous voilà commandant des troupes de magiciens, combattants et autre nains. Votre mission est de défendre le peuple Madrigal et Westens des maléfiques Soulblighter. Vous menerez vos troupes au combat en contrôlant chacun de leurs déplacements grâce à une interface 3D pilotée entièrement à la souris. Une carte d'accélération 3D (3dfx) est fortement conseillée.

Réf. LI15

VALABLE JUSQU'A EPUISEMENT DE STOCK

30,34 € TTC — 199 F

Red Hat 7.2

Exploitez les meilleures applications disponibles sous Linux, et bénéficiez d'une documentation très complète. Inclus : 9 CD, 4 jeux Loki ▶ support web de 60 jours. ▶ Comprend 3 suites bureautique et nombreuses applications pour les développeurs ▶ support de l'USB ▶ prise en charge 3D améliorée avec XFree86 4.0.3

Réf. LI32

79,95 € TTC — 524,44 F

DEBIAN GNU/LINUX 2.1

(PC Intel)

Il s'agit d'une distribution Linux 100% libre. Elle se compose de 4 CDROMs (2 CD binaires + 2 CD sources) soit plus de 2200 packages. La mise à jour vers de prochaines versions se fait en tout simplicité et gratuitement via le serveur FTP officiel Debian

Réf. LI12

5,95 € TTC — 39 F

LINUX Nirvana

Confirmé

Nous avons testé, trié et sélectionné pour vous les meilleurs logiciels LINUX disponibles sur Internet (plus de 600 Mo). Beaucoup sont livrés avec leurs sources et couvrent des domaines aussi divers que la P.A.O., le dessin, la C.A.O., les éditeurs, les langages, les utilitaires, etc...avec de nombreuses documentations. **Réf. CS25**

4,42 € TTC — 29 F

GNU Collection

Confirmé

What is GNU? "Gnu is Not Unix"! Nous avons récupéré pour vous le maximum de logiciels sous licence GPL comme Emacs, Tex, GCC, Gzip sont donc gratuits et LIVRÉS avec leurs SOURCES. Ces applications sont disponibles pour de nombreux systèmes d'exploitation, vous trouverez forcément votre bonheur. **Réf. CS24**

4,42 € TTC — 29 F