

IP³

Internet : Principes, Pratique et Programmation

Marc Tommasi

Octobre 2000

The idea of networking is probably as old as telecommunications itself. Consider people living in the stone age, where drums may have been used to transmit messages between individuals. Suppose caveman A wants to invite caveman B for a game of hurling rocks at each other, but they live too far apart for B to hear A banging his drum. So what are A's options? He could 1)-walk over to B's place, 2)-get a bigger drum, or 3)-ask C, who lives halfway between them, to forward the message. The last is called networking.

The Network Administrators' Guide. Olaf Kirch

Copyright (c) 2000 Marc Tommasi.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation ; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table des matières

1 Réseaux locaux	2
1.1 Un réseau local (LAN) : pour quoi faire ?	2
1.2 Le matériel	2
1.3 Se connecter au réseau local	4
1.4 À partir de chez soi	4
1.5 Le logiciel	4
1.6 Système Client/Serveur ou réseau poste à poste	5
2 Internet	7
2.1 Services Internet	7
2.2 Technologie Internet — Introduction	8
2.3 Les adresses	9
2.4 Internet et le Client-Serveur	10
2.5 TCP/IP	11
2.5.1 Principes	11
2.5.2 Remarques et extensions	13
2.5.3 Configuration dans un réseau local	13
2.5.4 Configuration depuis chez soi	14
2.6 Moteurs de recherche	14
2.6.1 Constitution de l'annuaire	15
2.6.2 Formuler une question	15
2.6.3 Rechercher dans l'annuaire	15
2.6.4 Procédés techniques	16
2.6.5 Comment se faire connaître ?	16
2.7 Autres cookies, cache, proxy et firewall	16
2.7.1 Le cache du navigateur	16
2.7.2 Les cookies	17
2.7.3 Les pare-feu	17
2.7.4 Le serveur mandataire	17
3 La messagerie électronique	18
3.1 Adresses électroniques	18
3.2 Clients et serveurs de messagerie	18
3.3 Logiciels	19
3.4 Un message électronique	20
3.5 Configurer sa messagerie	21
3.6 Attachements MIME	22
3.6.1 MIME	22
3.6.2 Attachements	22
3.7 Icq, Irc, etc	23
3.7.1 Irc	23
3.7.2 Icq	25

4	Le langage HTML	26
4.1	Introduction	26
4.2	Principes	26
4.3	Images	29
4.4	Formulaires	29
4.5	Feuilles de style	31
5	Programmation sous Internet	33
5.1	Programmation sur un serveur WEB	33
5.1.1	Pourquoi?	33
5.1.2	Solutions	33
5.1.3	Introduction à PHP	34
5.2	Côté client	35
5.2.1	Plugins	36
5.2.2	Les formulaires	36
5.2.3	Applets	36
5.2.4	JavaScript et VBScript	36
6	PHP et Javascript	37
6.1	PHP	37
6.1.1	Échappement, constructions	37
6.1.2	Variables	37
6.1.3	Opérateurs	38
6.1.4	Structures de contrôle	38
6.1.5	Fonctions	38
6.1.6	Tableaux	39
6.1.7	Traitement de formulaires	40
6.1.8	Accès aux bases de données	41
6.1.9	Fonctions usuelles, exemples et exercices	42

Avertissement

Une version de ce document est disponible en téléchargement sur

<http://www.grappa.univ-lille3.fr/polys>

avec bien d'autres polys d'ailleurs. Des versions au format LaTeX sont bien entendu disponibles à l'adresse

<http://www.grappa.univ-lille3.fr/polys/IP3/ip3.tgz>

sous forme d'archive compressée avec les programmes gzip et tar. Enfin, une version pdf est aussi disponible :

<http://www.grappa.univ-lille3.fr/polys/IP3/ip3.pdf>

Marc Tommasi

Une petite histoire

Les premiers réseaux ne sont pas informatiques mais liés aux télécommunications. Les réseaux du télégraphe (à partir de 1850) puis du téléphone, puis du télétype sont les ancêtres des réseaux informatiques d'aujourd'hui et beaucoup de principes encore en cours leur sont dûs. Par exemple, le système de codage Morse préfigure bien des techniques de numérisation¹ actuelles. Le symbole **A**, est codé en Morse par « *trait court, trait long* » pour être par exemple transmis par le télégraphe. Aujourd'hui on peut dire que le principe reste le même mais c'est le code qui a changé : ce sera peut être son code ASCII (01000001) qui sera transmis sur les câbles électriques. Au niveau physique, les réseaux fonctionnent alors selon des règles similaires, un échange d'impulsions électriques sur un fil, plus ou moins longues pour le Morse, ou alternées pour les réseaux modernes. Lorsque sur le support de communication les informations à transmettre sont codées de la sorte, on parle de réseau numérique.

Le réseau téléphonique est régi par un principe différent. La voix est convertie par le microphone (une membrane qui bouge en fonction du mouvement d'air que forme le son) en signal électrique qui varie en hauteur et fréquence (c'est une modulation) selon le volume et la hauteur du son à transmettre. De l'autre côté, le signal électrique est transformé par le haut parleur (une nouvelle membrane qui fait l'opération inverse). C'est le principe du fameux téléphone des enfants construit avec une cordelette et deux pots de yaourt. On parle alors de réseau analogique en opposition aux réseaux numériques.

Mais dès les années 60, les ingénieurs ont voulu connecter des ordinateurs aux réseaux téléphoniques. Ils ont inventé pour cela un appareil capable de transformer un signal analogique (modulé) en un signal numérique (démodulé) et vice versa. C'est le *modem* ou modulateur/démodulateur, aujourd'hui bon marché et présent dans de nombreux foyers sous forme de minitel, ou encore en association avec des ordinateurs personnels.

Les premiers réseaux informatiques ont été constitués d'un ensemble de terminaux simples autour d'ordinateurs centraux puissants. Par terminaux simples, on entend des machines ayant un clavier et un écran mais aucune unité de calcul. Ils étaient incapables d'effectuer quoi que ce soit de façon autonome. Tout était dirigé par l'ordinateur central. On parle alors de relation *maître/esclave*. Cette organisation existe encore et retrouve même un certain succès après avoir été décriée dans les années 80/90. La plupart des terminaux avaient alors une liaison directe à l'ordinateur central, mais ce dernier restait accessible à plus grande distance à l'aide de modems. Le minitel fonctionne de cette manière encore de nos jours.

Un degré de difficulté a été franchi lorsqu'on a voulu faire *coopérer* les ordinateurs et non seulement relier un poste à un site central. Motivées par des raisons économiques mais aussi par le désir de mieux communiquer, les entreprises se sont équipées de *réseaux locaux*. Pour y parvenir, un effort de *normalisation* était nécessaire, aussi bien au niveau des câbles et des connecteurs que du codage des informations qui empruntent le réseau. Les réalisations des constructeurs des années 60/70 différaient trop pour envisager une coopération simple des ordinateurs. Les organisations comme IBM, Xerox ou le Département de Défense américaine (DoD) vont initier un mouvement de normes.

Aujourd'hui la plupart des ordinateurs (largement plus d'un sur deux) sont connectés à des réseaux locaux (LAN), eux-mêmes reliés à des réseaux urbains (MAN), puis des réseaux mondiaux (WAN).

¹c'est la transformation d'une information sous la forme de nombre

Chapitre 1

Réseaux locaux

Pour mieux comprendre comment Internet par exemple fonctionne, il est nécessaire de détailler dans un premier temps ce qu'est un réseau local. Un réseau local est limité à une partie de bâtiment, un étage.

1.1 Un réseau local (LAN) : pour quoi faire ?

Connecter des ordinateurs au sein d'un réseau local offre de nouvelles possibilités, appelés *services*, aux usagers. Les machines qui assurent les services sont appelées **serveurs** et les machines qui en profitent sont appelées **clients**.

Quels services peut-on confier à un serveur sur un réseau local ? Tous les services serait-on tenté de répondre. Quelques exemples vont permettre d'illustrer les possibilités offertes. Un serveur peut s'occuper de la gestion du télex et de la télécopie. Un serveur peut s'occuper de la gestion des périphériques coûteux (imprimante, scanner, ...). Un serveur peut s'occuper de la documentation interne en gérant une batterie de CD-ROM. Un serveur peut être dédié à l'archivage. Un serveur, particulièrement puissant, peut être utilisé pour certains calculs. Bien entendu, dans la plupart des entreprises ou organisations, le service d'accès aux fichiers et aux bases de données partagées constitue une des tâches privilégiées des serveurs. Il existe de plus des services spécifiques comme des logiciels de messagerie, par exemple.

1.2 Le matériel

Conceptuellement, un réseau est un ensemble d'ordinateurs et d'équipements informatiques, appelés noeuds et reliés entre-eux. Logiquement cette liaison suit une *topologie*, c'est-à-dire, une forme géométrique. Il existe différentes topologies possibles :

Le bus. C'est la topologie la plus simple, mais elle a des inconvénients. Il est nécessaire d'avoir des répéteurs lorsque le nombre d'ordinateurs augmente. Un problème sur le câble entraîne une panne du réseau.

L'anneau. C'est, en fait, un bus refermé sur lui-même.

L'étoile. Tous les ordinateurs sont reliés à un dispositif matériel central appelé *hub*. Chaque noeud est indépendant des autres.

Dans un réseau, chaque noeud est identifié par une adresse.

Physiquement, il existe beaucoup de moyens de relier des ordinateurs. Les fils coaxiaux, des paires de cuivre torsadées, blindées ou non, et même des ondes.

Ethernet est une norme (IEEE 802.3) qui définit les types de câble à employer, la façon de les relier et la méthode utilisée pour transmettre les données. D'autres normes existent (ATM, token ring, arcnets, par exemple), qui sont parfois complémentaires ou concurrentes. Nous ne parlerons, pour illustrer notre propos, que de la norme Ethernet car c'est la norme la plus répandue.

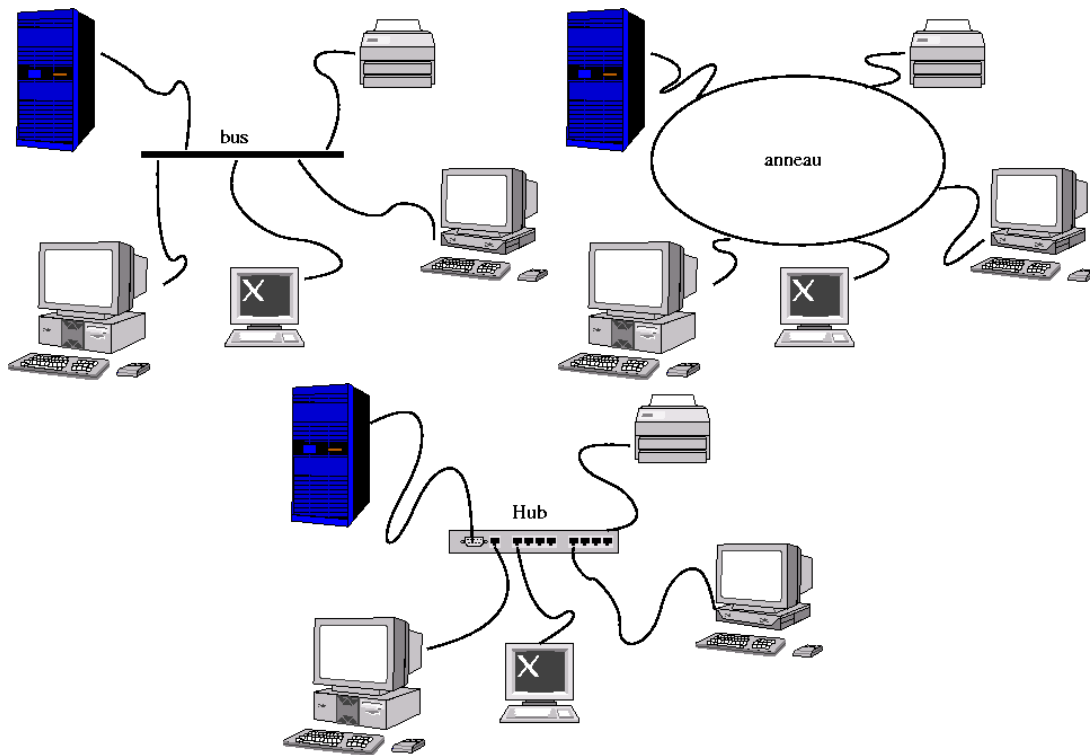


FIG. 1.1 – Topologie en bus, étoile et anneau

Les adresses sont fournies par le constructeur de l'équipement réseau. C'est donc un numéro « en dur », qui ne peut être modifié, unique au niveau mondial et donné dès la fabrication.

Les types de câble. Les trois types de câble sont : le coaxial épais, le coaxial fin et la paire torsadée. Le coaxial épais est utilisé pour relier des réseaux entre des bâtiments différents. Le coaxial fin est utilisé pour des petits réseaux (à partir d'une certaine taille, il faut utiliser des répéteurs). Le câblage en paire torsadée est actuellement le plus utilisé pour câbler les immeubles et bâtiments car moins cher et pouvant être utilisé pour le réseau téléphonique.

Les connexions. La topologie est en étoile ou en bus. Il faut aussi connecter le câble aux ordinateurs. L'ordinateur doit être muni d'une carte réseau compatible avec la norme ethernet, l'ordinateur et le câble. Les raccords entre la carte et le coaxial fin se font à l'aide de prises BNC, entre la carte et la paire torsadée à l'aide de prises RJ-45.

Les transmissions. La vitesse de transmission est de 10Mbps (1.2 million de caractères par seconde) ou 100 Mbps ; la méthode utilisée pour contrôler le flot d'informations est la méthode CSMA/CD (Carrier Sense Multiple Access / Collision Detection).

La vitesse de transmission est une vitesse théorique maximale et n'est pas souvent atteinte pour une raison qui s'explique en étudiant la méthode CSMA/CD. En effet, les ordinateurs pourront, après avoir testé s'ils sont bien connectés et que le support de transmission est actif (CS carrier sense), commencer à dialoguer et ceci à n'importe quel instant. Il n'y a pas de contrôle *a priori* et toutes les machines peuvent émettre en même temps. C'est la signification du MA pour multiple access. Le contrôle est effectué *a posteriori*. Si les éléments du réseau (cartes, hubs, ...) ont constaté une *collision* (CD), c'est-à-dire deux machines qui utilisent le support exactement en même temps, le signal est perturbé et l'information perdue. Une nouvelle émission est nécessaire. Lorsque le nombre d'ordinateurs reliés est faible, il y a peu de collisions et la vitesse de transfert des informations est proche de sa limite maximale. Par contre, lorsque le nombre d'ordinateurs

est grand, le réseau est très vite encombré et les performances chutent. Le réseau doit rester local pour rester utilisable.

Les autres protocoles, concurrents d'Ethernet fonctionnent différemment mais des limitations de taille comparables sont toujours présentes. Pour constituer de grands réseaux, il faut absolument agglomérer des réseaux plus petits.

1.3 Se connecter au réseau local

Pour se connecter, un ordinateur doit être équipé d'une carte réseau compatible avec la norme du réseau local. Insérée dans l'unité centrale de la machine ou directement intégrée à la carte mère, la carte ajoute un connecteur à l'ordinateur. Ensuite, le plus souvent un câble le relie à un hub. Le hub est un petit élément souvent placé dans une armoire avec de 3 à quelques dizaines de connecteurs. Il concentre les lignes. Les hubs sont parfois interconnectés ou reliés à des éléments actifs plus spécialisés : passerelles assurant le passage d'un réseau à un autre, switch permettant de diviser logiquement le réseau et améliorer ainsi les performances,...

1.4 À partir de chez soi

Lorsqu'on possède un ordinateur à la maison, et que l'on désire se connecter à un réseau (Internet ou un réseau d'entreprise) on ne peut plus utiliser un câble pour trouver le prochain hub. La solution consiste à utiliser les réseaux qui arrivent jusqu'à chez vous. Toutes les solutions ont été envisagées :

- le réseau électrique. Encore à l'état de projet, des tentatives ont été relatées dans la presse. Il ne semble pas que cela soit actuellement la solution la plus simple.
- le réseau téléphonique. C'est le plus simple car il est déjà utilisé pour le minitel. C'est un réseau analogique, un modem est nécessaire.
- le réseau du câble TV. Des expériences commerciales ont été lancées dans plusieurs villes en France et à l'étranger. C'est souvent déjà un réseau numérique, ce qui facilite la mise en fonction.

Dans le cas du réseau téléphonique, on peut noter l'évolution des technologies. Le minitel fonctionne avec un modem qui émet à la vitesse maximale de 75 bits par seconde (bps) et reçoit à 1200 bps. Les ingénieurs en télécommunications ont posé une borne supérieure pour la vitesse à laquelle il était possible d'utiliser les lignes : 28000 bps (28 kilo bps ou kbps). Grâce à des évolutions technologiques et des techniques de compression, les modems ont pu transmettre les données à 32 kbs, puis 56 kbps. Aujourd'hui avec les techniques DSL et ses dérivés (ADSL, XDSL...) on arrive à des hauts débits comme 1000 kbps (1 Mbps). Les lignes téléphoniques ont encore de beaux jours devant elles...

1.5 Le logiciel

Il existe toute une gamme de logiciels nécessaires au bon fonctionnement d'un réseau. Il est hors de propos dans cet exposé de présenter toute la complexité (due à la diversité des protocoles et aux différents niveaux) de tous ces logiciels. On parle de **Middleware** pour désigner l'ensemble des logiciels qui font croire à l'utilisateur (le client) que tous les serveurs locaux (et du monde) forment un système unique. Nous ne nous intéressons ici qu'au Middleware **NOS (Netware Operating System)**, i.e. aux systèmes d'exploitation réseau. C'est, en effet, la partie visible (en partie) pour l'utilisateur situé sur un poste client.

La connexion à un réseau local s'assimile à l'adjonction d'un nouveau périphérique. Ce périphérique est plus complexe que les périphériques locaux habituels et le système d'exploitation n'a pas toujours été conçu pour le prendre en compte. Il faut alors ajouter une couche au système d'exploitation pour prendre en compte le réseau ou utiliser un système d'exploitation de réseau. On peut, à l'aide de ce système, accéder à toutes les ressources disponibles sur le réseau (fichiers, programmes,

périphériques, ...). Par exemple, lancer une impression sur une imprimante distante se fait en choisissant l'imprimante puis en lançant une impression. L'utilisateur ne se préoccupe pas de la façon dont les données sont envoyées, ni comment elles transitent sur le réseau, ni comment l'imprimante les récupère, ni où se trouve l'imprimante (sauf pour aller récupérer le résultat de l'impression!).

L'une des fonctions du NOS est de rendre la localisation physique des ressources transparente aux applications. Dans les systèmes d'exploitation de réseau sont fournies des applications utilitaires : gestion des files d'attente d'impression, optimisation des accès pour les serveurs de fichiers, passerelles vers l'extérieur, progiciels de messagerie, systèmes de sauvegarde de fichiers à distance, procédures de sécurité, services de communication entre machines hétérogènes. La qualité de ces utilitaires conditionne souvent le choix du système d'exploitation. Les principaux systèmes d'exploitation intègrent des fonctions pour la prise en charge du réseau. Par contre, certains sont orientés pour équiper les serveurs :

UNIX/Linux issu du milieu informatique universitaire, UNIX est le creuset de l'industrie informatique. UNIX est disponible sur postes clients et sur serveurs. Il existe de nombreuses versions d'UNIX, ce qui nuit à son développement. UNIX est certainement le meilleur système généraliste. Mais, il est dépassé par des systèmes spécialisés dans certains domaines d'application. Les grands noms autour d'Unix sont SUN, HP, IBM, Digital. Linux est à l'origine la version d'Unix pour machines Intel. Il est maintenant disponible pour la plupart des plate-formes.

NT (Microsoft) plate-forme serveur développée par Microsoft et compatible avec Windows (et pour cause). Ce système est facile d'utilisation pour les programmeurs d'outils Microsoft. Ce système n'est pas le plus performant.

NETWARE (Novell) la plus importante base de serveurs dans les réseaux locaux a longtemps appartenu à Netware. Ce système permet d'avoir des serveurs de fichiers rapides et de bons serveurs de bases de données.

OS/2 (IBM) excellente plate forme serveur, une version existe pour les postes clients. OS/2 est une des plate-formes client-serveur les mieux gérées de l'industrie.

Les logiciels utilisés doivent être adaptés au réseau (logiciels spécialisés ou ajout de logiciels prenant en charge l'interface avec le réseau).

1.6 Système Client/Serveur ou réseau poste à poste

Il existe toute une hiérarchie de capacités pour les réseaux qui dépendent de nombreux paramètres et, en particulier, du système d'exploitation réseau sur chacun des postes. Nous prendrons les deux cas suivants : le cas du réseau poste à poste et le cas du réseau avec toutes les fonctionnalités d'un système d'exploitation réseau.

Réseau poste à poste C'est le cas d'un réseau de PC sous Mac ou WINDOWS 9x qui communiquent sans qu'il existe de système d'exploitation réseau. Dans cet environnement, des applications spécialisées permettent de partager des ressources qui se trouvent sur les différentes machines du réseau.

Les machines d'un réseau poste à poste sont typiquement des machines individuelles. C'est par exemple un réseau dans un secrétariat constitué de la machine du directeur, celle de sa secrétaire et celles de ses adjointes.

Prenons le cas où la ressource est un fichier. Des conditions doivent être réunies pour effectuer ce partage.

- La machine qui possède le fichier doit l'offrir en partage.
- La machine qui possède le fichier doit être allumée.

Le réseau poste à poste est limité car les gestions des utilisateurs, des fichiers et de la sécurité ne peuvent être uniformisées ni pilotées, ni administrées par un responsable. Ces fonctionnalités sont gérées indépendamment sur chaque poste et chaque logiciel du réseau.

Dans cette architecture, la sécurité et le contrôle sont quasi inexistantes. Aujourd'hui, La version poste à poste (peer to peer) est remise au goût du jour sur Internet. C'est selon ce principe que fonctionnent les désormais célèbres Gnutella et napster dédiés à la diffusion de données musicales.

Système d'exploitation réseau. Ce sont souvent des machines dont l'usage est partagé par plusieurs utilisateurs. Les machines sont banalisées.

Le système prend en charge le réseau et les applications utilisent le système pour communiquer. Cette organisation a beaucoup d'avantages pour l'utilisateur et les responsables du réseau.

- Les fonctionnalités de gestion des utilisateurs, des fichiers, de la sécurité sont globalisées sur le réseau et ne dépendent pas de la machine.
- Il est nécessaire de se connecter via une procédure appelée *login*, qui authentifie l'utilisateur.
- L'environnement de travail des utilisateurs est personnalisé, mais ne dépend pas de la machine utilisée : les machines sont *banalisées*.

Reprenons le cas où la ressource partagée est un fichier. Un utilisateur peut, sans donner d'instruction spéciale de partage, accéder à ce fichier quelle que soit la machine qu'il utilise. Cela est possible car l'utilisateur (et pas la machine) possède le fichier. Il peut, s'il le désire, l'offrir en partage à plusieurs utilisateurs. On peut retrouver une situation comme celle-ci : A peut modifier certains fichiers, que B ne peut que lire, et auxquels C ne peut pas accéder. Le système d'exploitation offre souvent la possibilité de définir des groupes d'utilisateurs ayant les mêmes droits sur les mêmes ressources.

Dans les deux cas, vous aurez, en général, à votre disposition des logiciels spécifiques tels que la messagerie et des accès vers l'extérieur (Internet par exemple).

Chapitre 2

Internet

Les grands réseaux (WAN = Wide Area Networks) ont été construits petit à petit par l'interconnexion de réseaux locaux. Le très médiatique Internet donne une bonne idée de ce que interconnexion et grand réseau signifient.

2.1 Services Internet

Avec les réseaux locaux, les utilisateurs tirent de nombreux avantages d'une interconnexion de machines : du partage des imprimantes et des matériels en général, des fichiers, . . . , jusqu'à la mise en place de services uniformes, simples et accessibles par tous, grâce aux outils client-serveur, de partage des informations de l'entreprise.

Avec les premiers pas de l'informatique, on a permis la gestion automatisée des informations de production, puis avec l'avènement des machines personnelles, l'entreprise s'est dotée de moyens puissants pour la bureautique. Avec le client-serveur, l'espace de l'informatique décisionnelle s'est créé. Maintenant grâce en partie aux technologies de l'internet, l'ère de l'informatique pour la *communication* s'ouvre.

Les services internet sont donc liés à la communication :

Échange de messages – e-mail La messagerie électronique (e-mail ou encore mail), c'est utiliser l'internet comme on utilise la poste. Il est possible de déposer un message dans la boîte aux lettres de son correspondant, qu'il soit ou non devant une machine. Ce dernier sera capable, à sa prochaine connexion, de consulter sa boîte aux lettres pour lire ou envoyer des messages à ses correspondants.

Les forums de discussion – News À l'inverse du mail où la discussion est réalisée de 1 à 1, de l'émetteur vers le destinataire, les forums de discussion (ou News) sont des moyens de discussion entre plusieurs personnes. La métaphore la plus précise est celle du kiosque à journaux, à la seule différence que tous les utilisateurs d'internet sont des journalistes potentiels. Tous sont capables de rédiger un article qui pourra être diffusé dans le groupe de discussion de son choix. Chacun peut lire le (ou s'abonner au) groupe de discussion qu'il désire.

Il existe des groupes de discussion sur tout et n'importe quoi. Des utilisateurs de Windows par exemple aux fanatiques des séries télévisées. C'est souvent dans ce cadre que se sont posées des questions éthiques quant à l'utilisation d'Internet, lorsque se forment des groupes au sujet par exemple du racisme ou de la pédophilie.

Échange de fichiers – Ftp Le service d'échange de fichiers permet de déposer des fichiers sur une machine distante, mais aussi, et c'est le plus fréquent de télécharger des fichiers sur sa machine. La distribution de logiciels gratuits, la diffusion d'images, de sons, de notes de cours pour les étudiants ou d'articles scientifiques sont parmi les utilisations les plus courantes de ce service.

La connexion à distance – Telnet Elle permet à un internaute (utilisateur d'internet) de se connecter et donc d'utiliser à distance une machine comme s'il se trouvait face à elle. Cela

ouvre par exemple des possibilités pour le travail à domicile, puisqu'il devient possible d'utiliser les machines sur son lieu de travail depuis chez soi.

Le Web – WWW Le service le plus connu, le plus récent et maintenant le plus utilisé de consultation d'hyperdocuments. Nous reviendrons dans la section suivante sur l'explication de ce qu'est un hyperdocument. C'est LE service d'internet. C'est ce service qui l'a rendu attrayant, et « commercialement utile ».

2.2 Technologie Internet — Introduction

D'un point de vue matériel, Internet n'est que la constitution d'un grand réseau, à partir de réseaux locaux. Tous les ordinateurs de ce grand réseau sont capables de « se parler ». Le protocole, donc la langue utilisée par les ordinateurs pour dialoguer, est TCP/IP (Transport Control Protocol/Internet Protocol).

Pour ajouter un ordinateur au réseau internet :

- Il faut avant tout posséder un dispositif de communication : une carte réseau si un réseau local existe ou un modem.
- Il faut ensuite obtenir d'une instance internationale décentralisée dans chaque pays (puis dans chaque réseau et chaque sous-réseau...) un numéro IP. C'est l'identifiant de la machine au même titre que le numéro de téléphone identifie un correspondant.
- Il faut que la machine soit capable de parler le TCP/IP. Cela implique d'ajouter au système d'exploitation des logiciels spécialisés. L'ensemble de tous ces logiciels est souvent appelé middleware car ils relient les logiciels applicatifs (SoftWare) au matériel (HardWare).
- Il faut ajouter les logiciels applicatifs pour utiliser les services de l'internet : pour lire le courrier électronique, participer aux groupes de discussion, ...

Ajoutons que les particuliers peuvent aussi se connecter à internet, bien qu'ils n'aient certainement pas de liaison directe chez eux. La solution est alors d'utiliser les lignes téléphoniques et un modem et de faire appel à un *fournisseur d'accès*. Ce sont des sociétés qui louent ou achètent un accès permanent à Internet et le louent. Ils offrent souvent divers services comme un courrier électronique, des possibilités d'hébergement d'hyperdocuments (ou pages) Web. Au début d'Internet, la tarification était basée sur un abonnement et un coût horaire, en plus des communications téléphoniques. Aujourd'hui presque tous sont gratuits.

Maintenant, pour dialoguer avec quelqu'un il faut savoir comment le nommer, l'appeler. On peut utiliser le numéro IP. Celui de ma machine est 194.254.132.55. Ce n'est pas très clair et difficile à retenir. D'autant plus que comme les numéros de téléphone, le nombre de chiffres (jusqu'à 12) risque d'encore augmenter dans les mois à venir. Il y a heureusement d'autres façons. En fait, les adresses de l'internet sont *domainisées*, c'est-à-dire, relatives à des domaines hiérarchisés. Pour le cas de ma machine, l'adresse domainisée est `13pc25.univ-lille3.fr` : elle se trouve en France (suffixe `fr`), à l'université de Lille 3 (`univ-lille3`) et s'appelle `13pc25`.

Le monde a donc été découpé en domaines hiérarchisés. Le « top level domain », le dernier suffixe dans le nom fait un premier découpage en zones. On utilise actuellement deux découpages : l'un thématique (`.net` réseaux, ou `.com` commerces ou `.org` organisations ...) l'autre géographique, (`fr` pour France, `de` pour Allemagne, `au` pour Autriche, `jp` pour Japon...).

Les sous-domaines sont souvent la désignation des entreprises ou des institutions (`univ-lille3` ou `sncf`, ...). Les noms des machines qui apparaissent dans le préfixe des adresses domainisées sont souvent nommées par le service qu'elles proposent : `www.sncf.fr` est la machine qui permet de lire des pages hypertextes à la SNCF en France tandis que ou `ftp.univ-lille1.fr` est la machine qui permet de faire de l'échange de fichiers à l'université de Lille 1 en France.

Pour que cela marche, internet comprend des machines qui contiennent une table qui associe numéros IP et adresses domainisées. Ces machines sont nommées DNS pour *domain name server*. Certains messages d'erreur courants font référence à cette table : *This address has not a DNS entry* signifie qu'une adresse ne peut être associée à aucun numéro IP.

2.3 Les adresses

Dans les années 90, un nouveau service de l'internet est apparu : le World Wide Web, la toile d'araignée mondiale encore désignée par l'acronyme WWW ou le diminutif Web. C'est ce service qui assure un certain succès à l'internet. L'idée est de lire des hyperdocuments à l'aide d'un navigateur.

Un hyperdocument est un document électronique contenant des images, du son, du texte, parfois des petits morceaux de programme, mais surtout des liens vers d'autres hyperdocuments : des liens hypertexte. Ces liens apparaissent dans un style qui les distingue, et une simple action de la souris sur un lien suffit à télécharger et présenter le document lié. Les documents peuvent se trouver sur n'importe quelle machine (serveur) de l'internet à des endroits parfois très éloignés et c'est ce qui donne l'impression à l'utilisateur de *naviguer* sur le réseau.

Le navigateur est l'outil qui permet de lire les hyperdocuments. On l'appelle aussi *browser* ou par le nom poétique de *butineur* et les deux plus connus aujourd'hui sont MicroSoft Internet Explorer (MSIE) et Netscape. Au début conçu pour ne lire que les hyperdocuments, le navigateur intègre aujourd'hui tous les services de l'internet (e-mail, ftp,...) et se substitue même au gestionnaire de fichier du système d'exploitation.

Le navigateur désigne par une adresse URL (Uniform Resource Locator), les adresses complètes de l'internet. C'est une adresse qui contient à la fois le nom d'une machine mais aussi le nom du service demandé, le nom d'un document, ... Voici des exemples de format d'adresse URL par type de service.

http

`http://machine/dossier/.../dossier/document`

C'est l'adresse d'un document hypertexte (dans l'arborescence) sur une machine dont le nom est donné par une adresse domainisée. Exemples :

- `http://www.univ-lille3.fr`. Aucun document précis n'est demandé : le document par défaut est désigné. C'est ce qu'on appelle la *page d'accueil*.
- `http://www.lifl.fr/LIFL/lifl.html`. Le document hypertexte `lifl.html` dans le dossier LIFL sur la machine `www.lifl.fr` est désigné.
- `http://www.lifl.fr/LIFL/lifl.html#equipe`. Le même document est demandé mais avec la précision d'un endroit particulier dans le document, repéré par un nom, ici `equipe`. Le mot `equipe` ne sera pas visible et reste une information interne cachée. C'est comme un signet dans un livre.

ftp

`ftp://utilisateur:motdepasse/machine/dossier/.../dossier/fichier`

On désigne le fichier ainsi que sa localisation dans l'arborescence de la machine. La machine est donnée par son adresse domainisée. Dans le cas où l'utilisateur n'est pas précisé, l'échange de fichier sera anonyme (anonymous ftp dans le jargon anglo-saxon) et les droits et les autorisations relatifs à la copie ou la lecture des fichiers seront limités. Dans le cas où l'utilisateur est précisé, on hérite des droits et des autorisations de l'utilisateur. Bien sûr, ce nom doit être présent dans l'annuaire de la machine (c'est quelqu'un de connu).

Exemples :

- `ftp://ftp.netscape.com/pub/communicator`. Cette connexion est anonyme.
- `ftp://tommasi:*/messaging.lifl.fr/home/infoto/tommasi/toto.doc` Je suis connu sur cette machine, mais je ne vous donnerais pas mon mot de passe ! J'ai plus de droits qu'un utilisateur anonyme, souvent celui de télécharger des documents *sur* le serveur (*upload*).

news

`news:nom.de.la.news`

Les groupes de discussions sont aussi hiérarchisés pour mieux s'y retrouver. Par exemple `news:comp.theory`, est un groupe qui parle de théorie dans la catégorie des groupes qui

parlent d'informatique (comp = computer science). Autre exemple `news :nord-pdc.annonces` est le groupe des petites annonces des gens du Nord-Pas-de-Calais.

mailto

`mailto :utilisateur@domaine.domaine.topdomaine`

Le @ se prononce souvent « at » pour dire « chez ». Par exemple je vais envoyer un mail à William qui travaille chez minimou, l'adresse est `mailto :William@minimou.fr`.

telnet

`telnet :utilisateur :motdepasse@machine`

Sous réserve que l'utilisateur soit connu de la machine, il ouvre une connexion à distance lui permettant de travailler comme si il se trouvait en face.

Enfin, la norme des URL a maintenant dépassé le cadre d'Internet. Dans les systèmes modernes on désigne aussi à l'aide d'une URL des ressources locales à la machine, sans que le réseau soit concerné. Par exemple : `file :/home/tommasi/mondocument.txt` désigne le document `mondocument.txt` qui se trouve sur ma machine dans le dossier `home/tommasi`.

2.4 Internet et le Client-Serveur

Dans Internet la notion de client-serveur est sous-jacente. Mais réciproquement, on assiste aujourd'hui à une standardisation du client-serveur grâce à l'Internet. Le paradigme du client-serveur s'applique totalement à l'Internet. On parle de service, de serveurs Web, de serveurs ftp... et de client Internet pour le navigateur.

Dans un premier temps, on peut distinguer dans le réseau Internet deux types de machines :

- celles qui ne servent qu'à consulter les informations (les clients), sur lesquelles n'existe qu'un navigateur. Elles ne font qu'utiliser les services de l'Internet.
- celles qui diffusent des services (les serveurs), des informations, etc.

Les machines utilisent le « langage » (protocole) TCP/IP pour dialoguer, et pour chaque type de service un « dialecte » (aussi nommé protocole) : HTTP (Hypertext Transfert Protocol) pour les hyperdocuments, NNTP (News Network Protocol) pour les news, SMTP (Standard Mail Transfert Protocol) pour l'e-mail, ...

Dans le cas des hyperdocuments, un dialogue standard peut être représenté comme dans la figure 2.1. Nous ne détaillerons pas la manière selon laquelle les messages sont transportés au sein du réseau entre le client et le serveur. Sachez simplement que les données à transmettre sont dans un premier temps découpées en un grand nombre de petits messages (ou paquets) et réassemblées à la fin du transfert.

Pourquoi se compliquer la vie en découpant d'abord le message ? La raison est à chercher dans l'efficacité d'utilisation de la ligne. Lorsque vous téléphonez chez quelqu'un vous avez le risque de trouver la ligne occupée. Vous devez alors attendre qu'elle se libère. Le protocole du téléphone est conçu de la sorte. On ne peut pas (c'est en train de changer) se partager la ligne parce que les informations sont transmises en flux continu. Cette situation n'est pas acceptable sur un réseau, les temps d'attente seraient trop longs. Découper les informations en paquet permet de réduire le temps de d'occupation des lignes et donc d'offrir la possibilité à plusieurs noeuds de discuter « presque » en même temps.

Ceci explique aussi assez bien le comportement des navigateurs : l'affichage d'un document peut se « bloquer » quelques instants (tous les paquets n'arrivent pas en même temps, et il y a des retardataires) ; ils affichent souvent le pourcentage de paquets reçus pour informer l'utilisateur et l'aider à patienter. Les façons de découper et de transmettre ne sont pas importantes, sachez seulement qu'elles suivent les protocoles TCP/IP et HTTP.

En étudiant mieux le langage HTML, vous remarquerez qu'une image n'est jamais incluse dans un document. C'est simplement une référence sous forme de nom de fichier qui est donnée, comme par exemple :

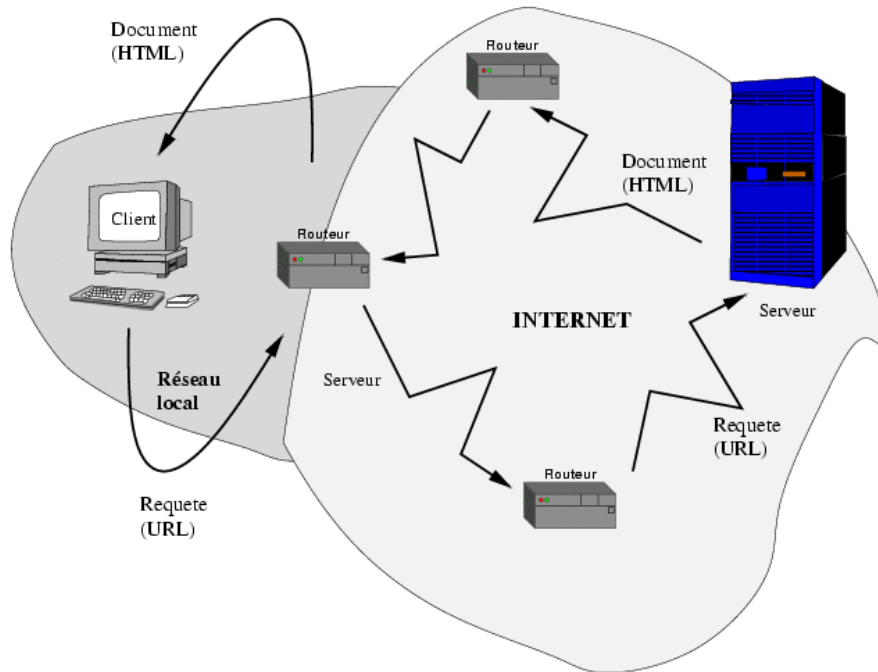


FIG. 2.1 – Dialogue standard entre un client et un serveur Web

```
<IMG SRC="http://www.grappa.univ-lille3.fr/icons/penguin.gif">
```

Dans cet exemple, l'image en question est contenue dans un fichier à part nommé `penguin.gif`. L'image est alors acheminée entre le client et le serveur indépendamment du reste de la page. C'est aussi cette méthode qu'on utilise pour le transfert de petits programmes, les applets (voir la section 5). Chaque inclusion de la sorte (image, applet,...) va déclencher autant de requêtes entre le client et le serveur.

Dans ce type d'utilisation, l'internet est vu comme un grand service de fichiers décentralisé. Mais nous verrons dans les sections programmation et tendances de l'internet les aspects liés à d'autres services.

2.5 TCP/IP

Cette section est un peu plus technique. Elle présente succinctement les principes techniques sous-jacents d'Internet. Ceci devrait contribuer à une meilleure compréhension du réseau et de sa configuration. Vous pouvez allègrement la passer si ce genre de chose ne vous concerne pas.

TCP/IP est le protocole de communication sur Internet. Son utilisation nécessite de disposer d'une première « couche logicielle et matérielle » de prise en charge du réseau plus basique comme Ethernet ou ATM ou Token Ring. La norme TCP/IP fournit des outils pour mettre en oeuvre de nouveaux protocoles pour cette fois faire fonctionner des applications comme le courrier électronique, l'échange de pages Web...

2.5.1 Principes

Transférer un document contenant du son et des images depuis une machine en Californie vers un ordinateur portable dans un taxi au milieu des embouteillages de Tokyo semble aujourd'hui

quelque chose de presque naturel pour beaucoup. Pourtant, je trouve cela toujours aussi surprenant. C'est qu'en fait, il y a de nombreuses étapes à comprendre et expliquer entre ce click de souris qui provoque une certaine agitation d'électrons et la transmission des informations. La prise en charge d'un réseau par un ordinateur passe par de nombreuses étapes qui vont de

1. l'utilisation d'une application réseau comme le navigateur, reposant sur
2. un protocole applicatif comme celui d'échange de documents hypertexte comme HTTP reposant sur
3. la résolution d'adresses domainisées Internet en numéros Internet puis sur
4. la fonction de routage sachant acheminer des informations à travers une interconnexion de réseaux vers le bon réseau local connaissant le numéro du destinataire puis sur
5. un adressage logique au sein du réseau local qui repose lui aussi sur
6. un adressage physique, transformation d'un numéro logique comme le numéro IP en une adresse physique comme le numéro Ethernet par exemple,
7. un protocole comme ethernet d'utilisation des supports physiques,
8. une transformation de valeurs numériques en impulsions électriques par exemple pour utiliser les câbles côté serveur ou encore en ondes hertziennes pour atteindre un portable.

Cette présentation est une tentative d'explication de l'expression **modèle en couches** que l'on rencontre dans de nombreux livres traitant de la question des réseaux. TCP/IP prend en charge une partie de ces étapes ou couches : les items 3 à 6 de cette liste soit les items 2 et 3 dans la dénomination officielle suivante :

1. Couche accès réseau
2. Couche internet (IP)
3. Couche transport (TCP/UDP)
4. Couche application (HTTP, SMTP, NNTP,...)

La couche application n'est pas concernée par TCP/IP. Elle l'utilise uniquement pour mener à bien ses tâches. La couche application est réalisée dans les logiciels à disposition des utilisateurs d'une machine comme un navigateur, un serveur Web ou un logiciel de lecture des news.

Prenons le cas de la transmission d'informations suivant le protocole HTTP dédié aux pages hypertexte. Nous avons vu que le client exprime sa requête sous la forme d'URL et que le serveur retourne le document demandé. HTTP décrit certes plus précisément ce dialogue mais en aucune façon les détails de la transmission elle-même des données n'est évoquée. Ce « détail » au niveau de HTTP sera réglé en demandant l'« intervention » de la couche transport.

L'application qui veut envoyer des données va communiquer avec la couche transport (**UDP** ou **TCP**) via un **port**¹. Elle choisira **UDP** (rapide mais pas très fiable) dans certains cas ou alors **TCP** (plus fiable et plus lent) dans d'autres. Les données sont transmises par paquets (assez petits dans le cas de TCP). La couche transport est chargée de l'établissement d'une connexion et du contrôle du flux de transmission.

Pour mener à bien ce contrôle, les données seront enrichies : des entêtes sont ajoutées aux paquets. Ensuite, c'est la couche Internet (**IP**) qui se charge de l'adressage, le routage à travers l'interconnexion des réseaux. Grâce aux adresses logiques (les adresses IP) accompagnant les données, IP détermine si l'adressage est local ou distant (en comparant avec sa propre adresse, mais surtout le **masque** du réseau). Dans le cas d'un adressage local, il fait appel à la couche suivante (accès réseau) ou alors transmet les données au réseau voisin via la **passerelle** ou **gateway** en anglais.

Encore une fois, les données sont enrichies pour assurer le routage : un nouvel entête est ajouté.

La couche accès réseau ne fait pas partie du protocole TCP/IP. C'est par exemple la norme Ethernet très répandue dans les réseaux locaux ou FDDI pour les fibres optiques, ATM pour les supports de réseaux métropolitains. La couche doit se charger de l'interface avec la carte

¹Le port est un numéro généralement associé, réservé à un service. Il peut même apparaître dans les URLs. Par exemple, le port associé au service HTTP est souvent 80. Essayez donc `www.grappa.univ-lille3.fr:80`.

réseau, suivre les règles du transport physique, composer les paquets, contrôler les erreurs de transmission. Le contrôle est réalisé en ajoutant un nouvel entête aux paquets contenant des informations spécifiques et en envoyant des accusés de réception.

La nature des informations spécifiques au contrôle est variable selon le protocole. Vous pouvez imaginer que la taille des trames, ou la parité de la somme des valeurs transmises ou tout autre description (partielle) des données pourrait être utilisée.

2.5.2 Remarques et extensions

Enfin, je voudrais terminer cette partie technique en ajoutant deux remarques. La première concerne le contrôle, la seconde les évolutions de TCP/IP.

La motivation du protocole d'Internet est d'interconnecter des réseaux aux protocoles physiques distincts (Ethernet, ATM, FDDI, Token Ring). Certains ne peuvent transporter que des données codées sur 7 bits (128 codes distincts) d'autres admettent des données codées sur 8 bits (256 codes distincts). Entre tous ces codages, une chose est respectée : le code est toujours le même pour les lettres alphabétiques minuscules et majuscules non accentuées, les chiffres et autres caractères typographiques usuels (retour à la ligne, virgule, point...). C'était suffisant pour les ingénieurs anglo-saxons dans les années 60/70. Aujourd'hui le réseau est planétaire et ces choix initiaux pèsent lourd pour les chinois, japonais, russes et européens avec leur lettres d'un autre alphabet (quand il y en a) ou même accentuées.

La première conséquence est que tous les protocoles ou langages fonctionnent avec des informations qui peuvent être codées sur 7 bits uniquement. C'est le cas du langage HTML par exemple qui ne se base que les caractères standards. Les autres caractères sont donc codés autrement : exemple, le code `é` pour désigner le é.

Le protocole ne gère pas une éventuelle perte d'information qui pourrait être due à un passage par des réseaux 7 bits. Il arrivait souvent et il arrive encore qu'un message électronique arrive à son destinataire avec des i remplaçant tous les é. Il a donc fallu décrire des normes supplémentaires comme *MIME*, au dessus d'Internet et des applications de base pour transporter ces informations plus riches. Nous en reparlerons dans la section dédiée aux attachements (cf page 3.6.2).

La deuxième remarque concerne donc les évolutions d'Internet. Avec la croissance du nombre de postes connectés à Internet, la limite des 4 milliards d'adresses va vite être dépassée. Les ordinateurs, les téléphones, agendas personnels, les voitures, les réfrigérateurs même auront un numéro IP. Une nouvelle génération de normes se prépare et TCP/IP sera remplacée par *IPV6*. On se doute que la transition sera difficile, certainement plus difficile encore que le passage à l'an 2000 ou l'euro si l'on tarde trop.

2.5.3 Configuration dans un réseau local

Pour ajouter une machine dans un réseau utilisant TCP/IP, il faut signaler quelle est son adresse IP, son masque de sous-réseau, l'adresse IP de sa DNS, celle de sa passerelle.

L'adresse IP Elle est composée de 4 nombres de 0 à 255 offrant donc 4 milliards de possibilités. Comme votre numéro de téléphone, elle doit être unique sur le réseau pour que le fonctionnement soit possible. Par exemple, le numéro IP de ma machine est 194.254.132.193.

Toutes les adresses IP ne sont pas routées par la couche Internet. Certaines restent dans le réseau local et n'en sortiront jamais, quelque soit le réseau en question. Par exemple, les numéros commençant par 194.254.132. sont routés alors que ceux commençant par 192.168.1. ne le sont pas² Toutefois, les machines possédant un tel numéro peuvent elles aussi interroger des machines distantes, mais en passant par l'intermédiaire d'un ordinateur spécialisé du réseau local qui lui possède une adresse routée. Ce service s'appelle le *IP masquerade*.

Cette possibilité permet de définir des intranets, de dépasser la limite des 4 milliards de machines, ...

²Vous pouvez considérer que les numéros non routés ont été choisis arbitrairement lors de la définition de la norme dans les années 60/70 même si ce n'est pas tout à fait vrai!

Parfois les numéros IP ne sont pas attribués de façon définitive à une machine, mais plutôt dynamiquement, à la demande, c'est-à-dire quand une machine en a besoin. C'est quasiment toujours le cas lorsque on utilise les services d'un fournisseur d'accès.

La DNS Elle est nécessaire pour beaucoup de protocoles applicatifs. Le service de DNS est assuré par une machine qui peut être sur un autre réseau que le réseau local. Elle est aussi désignée par son numéro IP.

Le masque de sous-réseau Il indique la classe du réseau qui sert à déterminer si une adresse sera locale ou distante. Par exemple si le masque est 255.255.255.0 alors l'adresse est dite de classe C et ce sont les trois premiers nombres qui désignent le numéro du réseau. Une adresse IP 194.254.132.192 avec le masque 255.255.255.0 désigne la machine 192 du réseau 194.254.132. Il y a au plus 255 possibilités pour les numéros IP de ce réseau donc au plus 255 machines sur ce réseau (en fait on réserve le numéro 0 pour désigner le réseau tout entier et 254 pour la passerelle). L'adresse 194.254.132.45 fait donc partie du même réseau. À l'inverse de l'adresse 194.254.131.131 n'en fait pas partie.

Il existe des réseaux de classe A (255.0.0.0), B (255.255.0.0) ou C.

La passerelle permet le routage vers les autres réseaux. C'est donc elle qui permet de faire sortir les informations. Elle est désignée encore par son numéro IP.

2.5.4 Configuration depuis chez soi

Chez soi, le problème est différent puisque vous avez fort peu de chances de vous trouver sur un réseau local. Vous n'aurez certainement pas non plus la possibilité de vous reposer sur Ethernet. Vous utiliserez alors certainement **PPP** pour couche d'accès au réseau via un modem et les protocoles d'authentification qui l'accompagnent (**CHAP** ou **PAP**).

Le protocole PPP signifie *point to point protocol*. Il permet de relier un point du réseau à un autre en passant par les lignes téléphoniques. L'établissement d'une liaison se fait exactement de la même façon que lorsque vous téléphonez : votre ordinateur va composer un numéro de téléphone, c'est un autre qui va « décrocher » de l'autre côté³.

C'est typiquement le moyen que vous offre votre fournisseur d'accès. Pour contrôler que vous êtes bien autorisés à établir une connexion avec le réseau distant, vous devez montrer « patte blanche ». C'est l'objet des protocoles PAP et CHAP. Ils sont basés comme à l'habitude sur la définition d'un nom de connexion et d'un mot de passe.

Pour configurer une liaison PPP, vous devez donc donner le numéro de téléphone de votre correspondant, un nom de connexion et un mot de passe.

Configuration TCP/IP Une fois le protocole PPP installé, vous pouvez utiliser TCP/IP. La configuration est la même que dans le cas d'un réseau local. Souvent, vous utilisez une passerelle par défaut (celle dont le numéro se termine par .254) et un numéro IP vous sera attribué dynamiquement.

2.6 Moteurs de recherche

Un **moteur de recherche** permet de fournir à l'internaute un annuaire des sites internet à la façon de l'annuaire des pages jaunes pour le téléphone. L'utilisateur accède à cet annuaire en formulant une question. Le moteur de recherche se charge alors de chercher dans son annuaire les pages qui peuvent répondre à sa question.

À partir de ces simples constatations, plusieurs questions peuvent être soulevées :

- Comment le moteur de recherche constitue-t-il son annuaire ?
- Comment l'internaute formule-t-il sa question ?
- Comment rechercher dans l'annuaire ?
- Quels procédés techniques sont mis en oeuvre ?
- Comment faire apparaître son site dans un annuaire ?

³Ne vous étonnez pas, les fax et les telex font ça depuis longtemps

2.6.1 Constitution de l'annuaire

Il existe deux façons de créer un annuaire Internet pour un moteur de recherche. Pour de nombreux moteurs, les deux techniques sont utilisées.

À la main C'est la façon la plus évidente. L'annuaire appartient à une société qui emploie des internautes professionnels qui se chargeront de parcourir le Web, de relever les pages intéressantes et de les mémoriser dans l'annuaire. Ils sont chargés aussi de vérifier si les pages déjà référencées existent encore et donc d'assurer une certaine maintenance.

Mémoriser les pages dans l'annuaire signifie aussi les classer, les ordonner, les ranger... Une partie de ce travail peut être automatisée, mais la tâche de classer un document reste encore difficile à automatiser, c'est un sujet de recherche en informatique très actuel.

Des milliers de pages sont créées chaque jour. Dans ce genre d'annuaire, on imagine très bien que les pages seront assez bien catégorisées, mais souvent les catégories seront incomplètes, ou désuètes.

Automatiquement Un logiciel que l'on appelle un *robot* est chargé de « parcourir Internet⁴ » pour collecter le plus possible de pages. Elles sont toutes rassemblées dans l'annuaire.

Il est clair qu'avec cette méthode, énormément de pages seront ajoutées à l'annuaire, mais le plus difficile sera de les ranger puisque les procédures automatiques ne fonctionnent pas très bien. Tout reposera donc sur la procédure de recherche et la façon de formuler sa question.

Les moteurs de recherche comme <http://www.google.org> et <http://www.altavista.com> sont des moteurs version *automatique*. À l'inverse, <http://www.yahoo.com> construit ses bases plutôt *à la main*. Ces indications sont assez grossières puisque comme indiqué précédemment, les deux techniques sont souvent utilisées en même temps.

2.6.2 Formuler une question

Prenons le cas où l'internaute cherche des pages traitant de *Miles Davis*. S'il choisit un annuaire où les pages sont rangées par thème, il s'aventurera dans une sorte d'aborescence par le moyen d'une suite de clics qui vont restreindre son espace de recherche. Par exemple : une première page proposera des catégories parmi lesquelles on trouvera *Arts*, puis dans la catégorie *Arts* on trouve *Musique*, puis *Style*, puis *Jazz* et *Interprètes* puis ...

Dans le cas d'un moteur automatique, l'internaute se trouve souvent face à un formulaire (cf. 5.2.2), c'est-à-dire une page dans laquelle se trouve une zone qui permet de saisir au clavier une suite de mots et un bouton (intitulé *Rechercher* ou *Submit* ou *Search* ou ...). Saisir *miles davis* dans cette zone et appuyer sur le bouton provoque l'affichage de plusieurs références de l'annuaire à des pages qui contiennent *à la fois* les mots *miles* et *davis*.

Il est donc probable que des pages n'ayant rien à voir avec le jazz soient dans cette liste. En effet les pages contenant une phrase comme *Je suis davis que t'as miles doigts dans la porte* par un débutant en Français ou encore *Davis University is five miles far from the airport...* contiennent aussi ces mots recherchés.

Les moteurs de recherche permettent souvent de raffiner une recherche. Un moyen très utile est de pouvoir exclure des mots. Par exemple *google* permet de dire *davis -university*, soit désigner les pages qui contiennent *davis* mais pas *university*.

Maintenant, la recherche doit être rapide, bien que l'annuaire puisse être très volumineux...

2.6.3 Rechercher dans l'annuaire

La technique qui permet d'effectuer une recherche rapide dans un ensemble s'appelle l'**indexation**. C'est la même technique que vous utilisez lorsque vous recherchez un index dans un livre ou un polycopié tel que celui-ci. Reste que la constitution d'un index est une affaire qui n'est pas si

⁴C'est une image. Le robot ne se déplace pas physiquement, mais ce n'est qu'une version automatisée de ce que vous pourriez faire à la main : suivre tous les liens que vous pouvez

simple, car il faut deviner comment le lecteur va s'en servir. Quels sont les mots à placer dans l'index ? Doit-il y avoir plusieurs index ? etc...

D'un point de vue informatique, la problématique est identique et les solutions sont diverses. La qualité d'un moteur de recherche dépend plus de la méthode d'indexation que du contenu de l'annuaire. C'est bien pour cela qu'existent des moteurs où le plus de choses possibles sont faites manuellement.

Vous pouvez en savoir plus sur ces méthodes en suivant les indications que les moteurs de recherche veulent bien vous donner pour assurer leur promotion.

2.6.4 Procédés techniques

Les annuaires sont rangés dans d'immenses bases de données. Les index sont des moyens d'accès rapide à ces données. Le serveur Web permet d'effectuer le traitement du formulaire et donc de la question du client et de transformer cela en une requête dans la base utilisant ces index.

Certaines méthodes sont jalousement gardées pour éviter à la concurrence de les utiliser. Cependant, il semble que la plupart soient assez classiques et souvent basées sur des calculs de fréquence d'apparition des mots, de téléchargement des pages etc...

La recherche d'information à travers le Web est un secteur en grande « ébullition » et de nombreux progrès restent à faire. Avec l'apparition de musiques, films et images sur le web, des champs d'applications et de recherche s'ouvrent continuellement.

2.6.5 Comment se faire connaître ?

Cette question dépend souvent du moteur de recherche. Pour les moteurs « manuels » il faut souvent se signaler en envoyant un message ou en remplissant un formulaire. Pour les autres, vous pouvez demander au robot de « passer » et aussi ajouter des éléments dans vos pages qui améliorent le classement.

Ce sont les balises META avec l'attribut `Keywords` directement présentes dans le code HTML qui permettent d'améliorer la classification automatique (puisque c'est le concepteur de la page qui indique les mots clefs du document).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Mon site sur Miles Davis</title>
<meta name="keywords" content="Miles Davis, Jazz, Musique, Interprètes">
</head>
<body>
....
</body>
</html>
```

2.7 Autres cookies, cache, proxy et firewall

Bien d'autres techniques sont mises en place dans le réseau Internet. Nous passons sous silence la plus utile qui devrait faire l'objet d'un cours entier : la sécurité avec ses déclinaisons comme l'authentification, la signature, le cryptage...

2.7.1 Le cache du navigateur

Pourquoi les liens visités sont-ils de couleur différente des liens non encore visités ? Comment fait le navigateur pour savoir cela ? Pourquoi votre navigateur va très vite lorsque vous appuyez sur la touche BACK ou PRÉCÉDENT ?

Votre navigateur utilise un **cache**, c'est-à-dire un tampon. Tous les documents déjà téléchargés sont enregistrés quelque part sur votre disque dur dans ce fameux tampon qui n'est qu'un simple dossier.

Le navigateur vérifie la présence d'un document d'abord dans ce tampon avant de le télécharger sur le WEB. Il peut contrôler aussi les dates des documents sur Internet et dans le tampon avant de se lancer dans un téléchargement.

Le tampon peut prendre beaucoup de place et il peut être utile de le vider parfois...

2.7.2 Les cookies

Les **cookies** sont une façon très simple d'espionner les internautes et occasionnent même parfois de sévères trous de sécurité. Même le gouvernement américain⁵ s'en émeut et incite toutes les administrations à ne plus les utiliser sur leurs serveurs certainement sous la pression de beaucoup d'internautes.

Techniquement, les cookies sont des informations (sans aucune restriction sur leur nature) que les serveurs WEB demandent au navigateur d'écrire dans un fichier sur le disque dur du client. Les cookies peuvent aussi être lus sur la demande des serveurs assurant ainsi le moyen de suivre les internautes à la trace.

Souvent les cookies sont activés par défaut et jamais un message ne vous avertit lorsque des cookies ont été enregistrés. Les utilisateurs novices ne se doutent même pas de leur existence. Il est donc nécessaire de fouiller dans les options (préférences) du navigateur pour les désactiver.

Le seul avantage que peuvent tirer les utilisateurs en acceptant les cookies est de trouver des pages personnalisées, en étant reconnus par les serveurs lors d'un nouveau passage. Cette technique est largement utilisée par exemple sur les sites de messagerie sur le Web comme HotMail, Caramail et consors : vous trouvez votre nom de connexion déjà présent dans les zones de saisies prévues pour l'authentification.

2.7.3 Les pare-feu

Ce sont des ordinateurs qui contribuent à sécuriser un réseau. Un **pare-feu** (*firewall* en anglais) est mis en place au sein d'un réseau d'entreprise dans le but de définir plus clairement ce qui est à l'intérieur et ce qui se trouve à l'extérieur du réseau d'entreprise. Le pare-feu est un passage obligé pour les informations qui doivent entrer ou sortir du réseau privé. Il joue alors le rôle de garde barrière en contrôlant la nature des informations qui transitent, les émetteurs ou récepteurs. Ils sont souvent couplés à un serveur mandataire pour des raisons d'efficacité.

2.7.4 Le serveur mandataire

Le **serveur mandataire** est la traduction française pour **proxy** server. Son rôle consiste à effectuer les requêtes vers Internet à la demande de ses clients, puis de leur retourner le résultat. Cet intermédiaire sert ensuite de gros tampon pour les documents Internet à l'instar du cache du navigateur WEB. Lorsque le nombre de clients est important, il permet souvent d'accélérer les échanges car les documents les plus demandés sont souvent déjà dans son tampon.

On retrouve souvent des serveurs mandataires sur les réseaux où de nombreuses machines clientes doivent passer par une unique machine pour accéder à Internet. C'est le cas pour les réseaux équipés de pare-feu et chez les fournisseurs d'accès à Internet.

⁵[ahrefurlhttp://www.wirednews.com/news/politics/0,1283,37233,00.html](http://www.wirednews.com/news/politics/0,1283,37233,00.html)

Chapitre 3

La messagerie électronique

L'échange de courriers électroniques est certainement l'un des plus vieux et des plus utilisés de tous les services sur Internet. Aujourd'hui de nouvelles techniques semblent concurrencer cet incontournable et différents moyens vous sont proposés pour en bénéficier.

Nous vous renvoyons à la section 2.1 pour une introduction à ce service. Nous détaillons maintenant son utilisation.

3.1 Adresses électroniques

Une adresse électronique est formée de la façon suivante :

`utilisateur@serveur.de.mail.`

Par exemple `tommasi@l3ux02.univ-lille3.fr.`

Il n'y a pas d'espace dans l'adresse et aucune distinction entre majuscules et minuscules n'est faite. Pour détacher l'adresse d'un utilisateur d'un serveur de messagerie particulier, il est possible de former des adresses en indiquant uniquement le nom de domaine de l'utilisateur. C'est alors la DNS (voir page 8) qui indique le nom du serveur du domaine. Mon adresse simplifiée devient alors : `tommasi@univ-lille3.fr.`

D'autres formes sont acceptées pour rendre la présentation de l'adresse plus agréable dans les outils de gestion de courrier électronique. Par exemple `Marc Tommasi <tommasi@univ-lille3.fr>` fonctionne encore.

3.2 Clients et serveurs de messagerie

Le dialogue qui s'installe entre les machines pour acheminer des pages hypertexte n'est pas le même dans le cas de la transmission de courriers électroniques. Dans le premier cas, il est nécessaire d'avoir une réponse immédiate, un vrai dialogue, le plus possible en temps réel. Dans le second, c'est une communication asynchrone entre deux individus. Le destinataire du message le lira quand il en aura envie. Le protocole mis en jeu dans l'échange de courrier tient bien-sûr compte de ces principes.

Prenons le cas où un individu *A* envoie un message à *B*. Leurs adresses sont `a@domaine_a.fr` et `b@domaine_b.fr`. Le dialogue est illustré en figure 3.1

Mme *A* compose son message sur sa machine et l'envoie à son serveur de messagerie. C'est celui-ci qui va acheminer au sein d'Internet le message de *A* vers le serveur de messagerie du domaine `domaine_b.fr`. La communication pour cet envoi suit le protocole **SMTP** (courriers sortants).

À l'arrivée, le serveur de messagerie de `domaine_b.fr` garde le message sur son disque dur. Il attend que Mme *B* consulte ses courriers. Elle pourra le faire le plus souvent avec le protocole **IMAP** ou **POP** (courriers entrants). Les nouveaux messages de *B* seront alors transférés sur le disque dur de son ordinateur (et disparaîtront certainement du serveur).

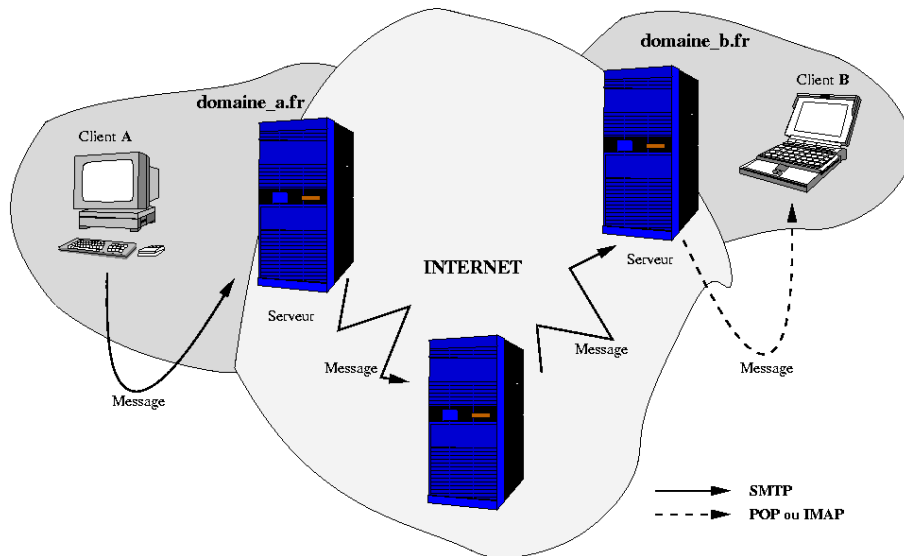


FIG. 3.1 – Dialogue client/serveur pour la messagerie

Plusieurs variantes existent. La première permet de se passer de POP ou IMAP lorsqu'on se trouve sur un réseau local et lorsque le disque dur du serveur de messagerie est partagé, grâce à un serveur de fichiers, sur tous les postes clients. On procède alors comme si on lisait ses messages directement sur le serveur. C'est souvent la solution choisie sur les réseaux Unix/Linux. Par expérience, je trouve que c'est la solution la plus pratique.

La seconde est souvent offerte sur les sites Internet qui offrent une messagerie électronique gratuite. La gestion des messages est accessible via le service HTTP (les pages hypertexte). C'est par exemple les services Hotmail, Caramail,...

avantages Vous utilisez toujours un navigateur WEB. Vous accédez à vos messages depuis n'importe quel poste. Vous n'avez pas besoin d'abonnement.

inconvénients – C'est finalement plus cher que prévu. Lorsque vous envoyez ou consultez votre messagerie de façon traditionnelle avec POP/SMTP, vous n'occupez qu'un court instant la ligne téléphonique ou le réseau. La gestion de vos courriers (lecture, rédaction, rangement) peut être faite sans connexion à Internet. Par contre, via HTTP, vous devez être *toujours* connecté.

- C'est peu pratique. À l'usage, lorsque vous avez beaucoup de courrier, vous vous apercevez que les outils sont lents, et lorsqu'un problème arrive vous êtes désarmés.
- Aucune sécurité. Les messages ne sont jamais sur votre ordinateur. Croisez les doigts pour qu'aucun problème ne survienne sur ces machines loin de chez vous... Le système HotMail (Microsoft) a été piraté. Toutes les boîtes aux lettres sont restées « ouvertes » à tous pendant plus d'un week-end. Une chance que les pirates aient été gentils de ne rien effacer...
- Aucune confidentialité. Les administrateurs des serveurs HTTP et SMTP/POP peuvent tout voir et tout faire. Lorsque cela reste dans votre société, c'est peut être acceptable. Ailleurs...

3.3 Logiciels

Il existe beaucoup, mais vraiment beaucoup d'outils de gestion des messages électroniques (on dit aussi *client de messagerie électronique*). Ceux-ci permettent de

Composer (Compose) des messages. C'est-à-dire, écrire un message en demandant les champs requis.

Envoyer (Send) des messages. Cette action est dissociée de la composition pour permettre aux utilisateurs qui ne bénéficient pas d'une liaison permanente à Internet de décaler dans le temps leurs envois. La fonction d'envoi de message met en relation le client de messagerie avec son serveur. Voir la section 3.2.

Lire et vérifier (Get/Read et Check) des messages. Le client doit être capable d'interroger un ou éventuellement plusieurs serveurs de messagerie pour vérifier la présence de messages et ensuite les télécharger.

Le serveur de messagerie marque ou supprime les messages téléchargés. Ces derniers ne seront donc plus accessibles, le plus souvent à la prochaine demande de lecture. Une fois téléchargés ils se trouvent sur le disque dur de la machine utilisant le client de messagerie.

Répondre (Reply) à un message. Les fonctions disponibles à ce sujet permettent de répondre à l'auteur en citant ou pas son texte original.

Transmettre (Forward) un message.

Ranger son courrier. C'est une action importante. Avec le temps de nombreux messages s'accumulent et il est souvent nécessaire de les retrouver facilement. On rencontre alors les mêmes solutions que pour l'organisation des documents sur un disque : dossiers, sous dossiers, etc... avec les procédures de suppression, déplacement, copie, édition habituelles.

Lorsque le courrier est toujours lu sur le même ordinateur, le gérer et le retrouver est une opération simple : tous les messages sont sur le même disque dur. Si l'utilisateur est « itinérant » et consulte sa messagerie sur de nombreux ordinateurs, sur des réseaux distincts comme par exemple chez lui, sur son téléphone ou à son travail, la gestion est plus difficile. Il existe plusieurs solutions pour cela.

3.4 Un message électronique

Un message électronique est un message transmis sur un réseau. À l'instar des courriers postaux traditionnels, un courrier électronique est placé à l'intérieur d'une enveloppe. Certains éléments figurant sur l'enveloppe sont ajoutés par le *serveur de messagerie* (*i.e.* le facteur) pour permettre l'acheminement, d'autres sont spécifiés par l'utilisateur. Les données apparaissant sur l'enveloppe sont souvent qualifiées de champs. J'ai reporté les noms des champs en anglais comme on les trouve dans l'enveloppe, bien que parfois ceux-ci soient parfois présentés en français dans certains logiciels.

- Le champ **From** est obligatoire et rempli automatiquement par les logiciels de lecture et composition de messages. Il contient l'adresse électronique de l'émetteur. C'est cette valeur qui sert comme adresse de réponse quand le champ **reply-to** n'est pas présent. Les champs suivants sont à remplir par l'émetteur d'un message.
- Le champ **reply-to** est facultatif et permet de spécifier l'adresse de réponse au message. Il peut être fort utile lorsque vous utilisez des adresses d'émission temporaires.
- Le champ **To** est obligatoire et contient l'adresse du ou des destinataires. Les adresses sont séparées par des virgules.
- Le champ **CC** est facultatif et permet d'ajouter des destinataires qui recevront une « copie conforme » du message.
- Le champ **BCC** a la même fonction que le champ **CC**, mais aucun des destinataires ne peut connaître la liste figurant dans ce champ **BCC**. C'est ce que l'on appelle une « copie conforme aveugle » (*Blind Carbon Copy*).
- Le champ **Subject** contient une information facultative pour indiquer le propos du message.
- Vient ensuite le **Corps** du message. Quelques restrictions existent pour le format des données qui s'y trouvent. Nous y reviendrons.

Bien d'autres champs apparaissent dans l'enveloppe, surtout quand le message est enfin arrivé à son destinataire. Ils sont ajoutés automatiquement. Certains d'entre eux sont donnés en exemple ci-dessous. Les noms des champs sont indiqués en tête de ligne et terminés par `:`.

```
From Jean-Luc.Dupont@lifl.fr Tue Oct 31 15:40:18 2000
Return-Path: <Jean-Luc.Dupont@lifl.fr>
Received: from l3av03.univ-lille3.fr (l3av03.univ-lille3.fr [194.254.131.200])
    by l3ux02.univ-lille3.fr (8.9.3/8.8.7) with ESMTP id PAA27201
    for <TOMMASI@L3UX02.UNIV-LILLE3.FR>; Tue, 31 Oct 2000 15:40:18 +0100
Received: from malonne.lifl.fr (134.206.10.29) by l3av03.univ-lille3.fr (MX
    V5.1-X AnDp) with ESMTP for <tommasi@univ-lille3.fr>;
    Tue, 31 Oct 2000 15:39:43 +0100
Received: from lifl.fr (velize.lifl.fr [134.206.10.235]) by malonne.lifl.fr
    (8.9.3/jtpda-5.3.3) with ESMTP id PAA23618 for
    <tommasi@univ-lille3.fr>; Tue, 31 Oct 2000 15:39:53 +0100 (MET)
Message-ID: <39FEDAB7.9994DE2A@lifl.fr>
X-Mailer: Mozilla 4.61 [en] (X11; I; Linux 2.2.12-20 i686)
X-Accept-Language: fr, en
MIME-Version: 1.0
References: <200010311257.NAA12266@l3mr193.univ-lille3.fr>
Content-Type: text/plain; charset=iso-8859-1
X-MIME-Autoconverted: from 8bit to quoted-printable by malonne.lifl.fr id
    PAA23618
Content-Transfer-Encoding: 8bit
X-MIME-Autoconverted: from quoted-printable to 8bit by l3ux02.univ-lille3.fr id PAA27201
Resent-Message-Id: <200010311440.PAA27201@l3ux02.univ-lille3.fr>
Resent-Date: Tue, 31 Oct 2000 15:39:44 +0100
Resent-From: tommasi@univ-lille3.fr
Resent-To: <TOMMASI@L3UX02.UNIV-LILLE3.FR>
From: Jean-Luc Dupont <Jean-Luc.Dupont@lifl.fr>
Sender: Jean-Luc.Dupont@lifl.fr
To: Marc Tommasi <tommasi@univ-lille3.fr>
Subject: Re: Encadrement
Date: Tue, 31 Oct 2000 15:44:07 +0100
```

Il est intéressant de noter que cette enveloppe montre que le message est passé par plusieurs serveurs de messagerie avant d'arriver à destination. Ce sont les champs `Received` qui mentionnent ce trajet.

3.5 Configurer sa messagerie

Sur le client Vous devez bien sûr spécifier votre nom et votre adresse email, et éventuellement une adresse de réponse. Puis, donnez un serveur de messagerie pour les courriers entrants avec le protocole de communication et la même chose pour les courriers sortants. Souvent les courriers entrants utilisent le protocole POP et les serveurs associés s'appellent `pop.xxx.yyy`. De même, les courriers sortants sont gérés par le protocole SMTP et les serveurs associés s'appellent `smtp.xxx.yyy`.

Ensuite, vous devez spécifier des noms d'utilisateur et un mot de passe pour ces serveurs¹.

Le reste c'est du décor. Je donne quelques exemples :

Vous pouvez demander à votre client de se souvenir de votre mot de passe. **Méfiez-vous** : n'utilisez jamais cette option lorsque vous utilisez une machine sans système d'exploitation multi-utilisateur (Windows 9x, Mac) accessible par de nombreuses personnes.

¹ *a priori* seul le serveur de courriers entrant demande cela, ce qui permet donc d'envoyer des messages sous le nom de quelqu'un sans que cela pose de réel problème technique...

Vous indiquerez aussi si les messages téléchargés sur votre machine doivent être détruits sur le serveur. C'est souvent le cas mais le contraire peut se révéler intéressant lorsque vous voyagez entre plusieurs réseaux (chez vous, au travail...)

Vous pouvez demander au client de vérifier périodiquement et pour vous la présence de nouveaux messages.

Enfin, pour les actifs des messageries, vous pouvez même demander à votre logiciel de ranger vos messages pour vous, automatiquement, en fonction de l'émetteur, du sujet, ou même de la présence de mots dans le corps.

Sur le serveur Après quelques changements de fournisseur d'accès, d'universités ou même d'emploi, vous avez 4 ou 5 adresses électroniques que vous avez allègrement distribuées à vos connaissances. Lire vos messages devient une épreuve de force car il faut vérifier toutes ces boîtes. Vous avez pu anticiper un peu en positionnant le champ `Reply-to`, mais ce n'est pas satisfaisant. Vous avez la possibilité de paramétrer votre serveur de messages pour que tout ce qui vous est adressé soit désormais transmis à une autre adresse, comme avec la poste lors d'un déménagement. Sous Unix/Linux c'est un simple document `.forward` qui indiquera la nouvelle adresse.

Vous partez en vacances pendant 6 mois dans les îles pacifiques. Vous voulez avertir vos correspondants que vous ne lirez pas vos messages avant longtemps. Un document `.vacation` permet de contenir le texte d'une réponse automatique.

3.6 Attachements MIME

Comment expédier un document à un destinataire ? La manière la plus simple est de le copier dans le corps du message. Malheureusement, beaucoup de documents contiennent des codes qui ne sont pas alphabétiques ou numériques. C'est le cas pour ceux composés avec des traitements de texte. La méthode d'acheminement des messages n'assure pas que ces codes ne seront pas altérés pendant leur transfert (voir section 2.5.2). Seuls les codes de caractères compris entre 0 et 127 sont sûrs de passer correctement à travers les réseaux. Les codes des caractères accentués ne sont pas dans cette plage. Pire, les formats obscurs de documents comme Word utilisent des codes de 0 à 255.

3.6.1 MIME

La solution envisagée a été de définir des normes d'encapsulation de documents : **MIME** pour *Multipurpose Internet Mail Extensions*. Un message peut donc être ajouté dans un document à condition d'être codé au format MIME. Vous avez le choix du codage entre **Base64** et **quoted-printable**. Le tableau suivant montre un exemple de codage :

texte		alors c'était ça ?
Base64		YWxvcnMgYyfpdGFpdCDnYSA/Cg==
quoted-printable		alors c'=E9tait =E7a ?

Vous remarquerez que dans les deux cas, tous les caractères accentués ont été remplacés. Le résultat d'un codage base64 n'est plus du tout lisible pour l'homme, mais peu importe. Ce sont les outils de messagerie qui se chargeront d'effectuer le codage et le décodage. L'utilisateur devrait ne jamais se rendre compte de ces manipulations. Malheureusement, ce n'est pas toujours le cas...

Il existe bien d'autres méthodes pour convertir des documents et les faire ainsi passer sur Internet. Aujourd'hui la norme MIME est largement diffusée et utilisée.

3.6.2 Attachements

Un **attachement** est une ou plusieurs pièces jointes à un message. L'intérêt de l'attachement est d'éviter à l'utilisateur de se trouver face à des messages dont les textes apparaissent comme

dans le tableau précédent. Un exemple d'attachement et de son rendu sont présentés en figures 3.2 et 3.3.

L'attachement est composé automatiquement par la plupart des clients de messagerie. Contrairement à ce que beaucoup peuvent penser, un attachement n'est pas acheminé indépendamment d'un message. Il est compris *dans* le message. Le reste n'est que présentation.

Vous remarquerez dans le texte réel du message plusieurs indications : `Content-Type : Multipart/Mixed ;` et `boundary` qui indiquent dans l'enveloppe que le message est en fait composé de plusieurs parties. Puis, `MIME-Version` qui indique qu'il est au format MIME. Enfin que les documents ont subi un encodage *quoted-printable* pour le premier et *Base64* pour le second.

De Marc Tommasi <tommasi@free.fr>
A : [tommas](#)

Marc Tommas
<http://www.grappa.univ-lille3.fr/~tommasi>

 [doc1](#)
ceci est un premier document

 [doc2](#)
Et voici un second

FIG. 3.2 – Rendu d'un attachement

3.7 Icq, Irc, etc

Le principe de l'email avec ses communications purement asynchrones a gêné certains utilisateurs. Il manquait un outil de communication synchrone comme le téléphone. Cet outil existe depuis longtemps sur beaucoup de systèmes comme Unix et s'appelle *talk*. Des dérivés ont vu le jour depuis. Ce sont les outils *icq*, *Irc* aussi connu sous le nom *chat* et aussi *instant messenger*.

3.7.1 Irc

Irc signifie *Internet Relay Chat* (d'où son diminutif plus connu). C'est en fait un « vieux » service qui date 88, successeur du service *talk* un peu désuet. Les utilisateurs de Irc se connectent sur des serveurs, dialoguent entre eux sur des canaux (*Channel*), peuvent s'envoyer des fichiers, etc.

La difficulté de l'Irc lorsqu'on débute est triple : trouver un serveur et un canal intéressant, comprendre les commandes pour envisager les possibilités offertes par ce service, comprendre les nombreuses abréviations employées par leurs usagers.

Malgré ces inconvénients qui peuvent vous effrayer, Irc reste un outil convivial pour la discussion de groupes, à bâton rompu, le tout au niveau planétaire.

From: Marc Tommasi <tommasi@free.fr>
Reply-To: tommasi@univ-lille3.fr
To: tommasi
Date: Mon, 30 Jul 2001 22:22:12 +0200
X-Mailer: KMail [version 1.0.29]
Content-Type: Multipart/Mixed;
 boundary="Boundary-=_nWlrBbmQBhCDarzOwKkYHIDdqSCD"
MIME-Version: 1.0
Message-Id: <01073022232700.15251@localhost.localdomain>
Status: RO
X-Status: Q

--Boundary-=_nWlrBbmQBhCDarzOwKkYHIDdqSCD
Content-Type: text/plain
Content-Transfer-Encoding: 8bit

--

Marc Tommasi
<http://www.grappa.univ-lille3.fr/~tommasi>

--Boundary-=_nWlrBbmQBhCDarzOwKkYHIDdqSCD
Content-Type: application/octet-stream;
 name="doc1"
Content-Transfer-Encoding: quoted-printable
Content-Description: ceci est un premier document
Content-Disposition: attachment; filename="doc1"

alors c'=
=E9tait =E7a ?

--Boundary-=_nWlrBbmQBhCDarzOwKkYHIDdqSCD
Content-Type: text/plain;
 name="doc2"
Content-Transfer-Encoding: base64
Content-Description: En voici un second
Content-Disposition: attachment; filename="doc2"

SidhaW1lIGxlcYBjaG9leCBjb21tZSB1biBmb3UK

--Boundary-=_nWlrBbmQBhCDarzOwKkYHIDdqSCD--

FIG. 3.3 – Code de l'attachement, c'est-à-dire le texte réel du message.

3.7.2 Icq

Icq (*I seek you*) est un outil dédié à un envoi de messages entre deux personnes à l'image du service mail. La différence est qu'il repose sur deux nouveaux principes : un annuaire et un état de connexion des usagers.

Les annuaires sont mémorisés sur des serveurs comme mirabilis. Les utilisateurs y sont répertoriés avec un nom, un pseudo. Ils peuvent ajouter plusieurs informations les concernant (adresse email ou web, adresse, pays, âge, sexe). La fonction de ces annuaires est d'offrir la possibilité de rechercher un utilisateur, de contrôler l'état de sa connexion et donc de le contacter.

L'état de connexion d'un utilisateur signale sa présence sur le réseau et assure donc que les messages pourront être lus immédiatement. Les états possibles sont : connecté ou déconnecté, ainsi que quelques variations (absent pour un instant, occupé, ...)

Chapitre 4

Le langage HTML

4.1 Introduction

HTML (*Hypertext Markup Language*) est langage de description de documents. C'est un langage particulier issu de la norme SGML (*Standardized Generalized Markup Language*) qui définit des langages de balisage. C'est le langage qui a été construit pour décrire les documents hypertexte sur Internet. HTML est normalisé par le W3C (consortium qui regroupe de nombreuses entreprises et organisations). Être normalisé signifie que les acteurs d'Internet se sont mis d'accord pour suivre tous les mêmes règles pour écrire et traiter des documents hypertexte sur le WEB.

Les origines de HTML sont diverses. La notion d'hypertexte date d'au moins les années 60. La norme SGML est elle aussi antérieure à Internet. La naissance d'HTML date de la fin des années 80 quand des ingénieurs du CERN et particulièrement Tim Berner Lee, ont développé le premier réseau d'échange de documents et le premier navigateur.

Les avantages du langage HTML sont nombreux. Il est simple à utiliser, sa conception lui permet de rester indépendant vis à vis des plate-formes et de pouvoir être échangé sur les réseaux. En effet, tout document HTML n'est écrit qu'avec les caractères alphabétiques standards. Ceci lui donne deux avantages. Premièrement, il peut être composé sur n'importe quel système avec un simple éditeur de textes. Aucun logiciel spécialisé n'est nécessaire pour composer des pages HTML. Deuxièmement, en évitant d'utiliser des codes différents des caractères alphabétiques standards, on écarte les problèmes de conflits avec les codes servant au transport des informations sur le réseau (codes de contrôle, d'encapsulation...).

Les principes d'HTML sont de donner dans le document des informations de structure du texte (titre, niveau de titre...), de présentation (mettre en gras, centrer,...) et ensuite les données elles-mêmes (le texte). La présentation (ou composition) du texte est donc à la charge du navigateur.

Les inconvénients du HTML se découvrent au fur et à mesure que les techniques d'Internet évoluent. Mais les évolutions du langage à travers ses normes successives (nous en sommes à la version 4 (datant de 97) du langage HTML) pallient peu à peu ses défauts. Le reproche le plus évident maintenant était justement de mélanger dans le document les aspects mise en forme, structure et données.

Le HTML 4 avec ses feuilles de style permet de dissocier données/structure et présentation. Le langage XML dont nous ne parlons pas ici permet d'identifier clairement les données.

4.2 Principes

HTML est un langage de balisage. Une balise (*Tag* en anglais) est une suite de mots sans distinction entre majuscules et minuscules entre < et > (*balise de début*) ou < / et > (*balise de fin*). La balise donne des informations de structure ou de présentation à un texte qu'elle encadre par la balise de début et la balise de fin. Par exemple, le paragraphe, comme élément de structuration du texte est compris entre la balise de début <P> et de celle de fin </P>.

Une balise peut avoir des attributs (*attribute* en anglais). Exemple pour un paragraphe centré :
<P ALIGN="center">tarata</P>.

La forme générale d'un document HTML est la suivante :

```
<html>
<head>
<title>...</title>
</head>
<body>
...
</body>
</html>
```

La balise `head` définit l'entête du document où l'on peut indiquer le titre qui sera utilisé dans la barre de titre du navigateur ou encore dans les signets. La balise `body` encadre le corps du document. Voyons quelques balises à travers un exemple. Le code se trouve en figure 4.1 et le rendu dans la figure 4.2.

```
<html>
<head>
<title>Ma page Web</title>
</head>
<body bgcolor="#FFFFFF">
<h1>Bonjour !</h1>
<p>je me présente :           <b>Eugène Poudrier</b>,
spécialiste en </p>
<ol>
<li> html
<li> internet et <li> intranet
</ol>
Voici ma photo : . Si vous voulez mieux me
connaître, cliquez <a href="http://www.eugene.org">ici</a>.
<hr>
<address>
<table>
<tr><td colspan="2">
    <a href="mailto:poudrier@eugene.org">Eugene Poudrier</a><BR>
    </td>
</tr>
<tr><td>
    Sté Poudrier and sons
    </td>
<td>
    Paris, Washington, Singapour
    </td>
</tr>
</table>
</address>
</body>
</html>
```

FIG. 4.1 – Code HTML

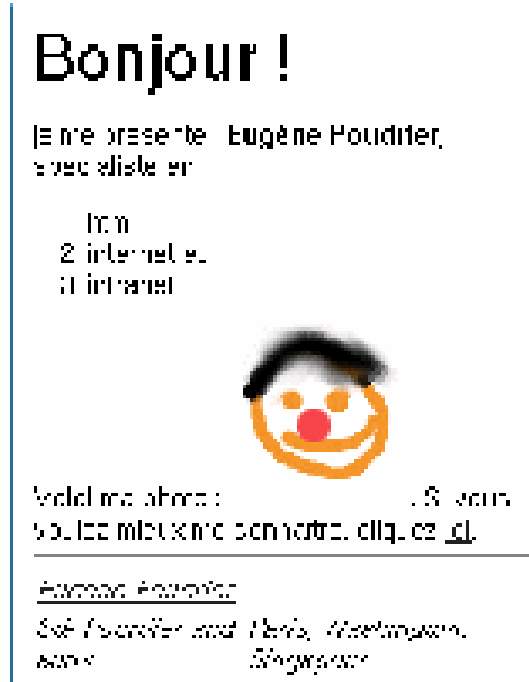


FIG. 4.2 – Rendu dans un navigateur

Le mot *Bonjour!* apparaît comme un titre de niveau 1. Quatre niveaux sont disponibles en HTML. La balise `` permet de passer en gras (*boldface* en anglais), mais ce qu’il est important de noter ici, c’est que les espaces ou retours à la ligne présents dans le document HTML ne sont pas reproduits dans le navigateur, ils n’ont qu’une seule signification : séparateur de mots. En d’autres termes, un espace, *n* espaces, un retour à la ligne, *n* retours à la ligne signifient toujours la même chose : changement de mot.

Certaines balises de fin ne sont pas nécessaires (par exemple ``). Il est tout de même préférable de les mettre dans un souci de clarté. Pour les listes numérotées (`OL`, *ordered list*), les numéros devant les éléments de liste (`LI list item`) sont calculés par le navigateur. Encore une fois, l’absence ou la présence de retours à la ligne est sans incidence sur le résultat.

Une image n’est jamais présente dans un document HTML. C’est seulement une référence (`IMG`) à un fichier contenant l’image qui apparaît. C’est à la charge du navigateur de demander ce fichier au serveur mentionné dans l’attribut `SRC` pour l’afficher au sein du texte. Le document et l’image peuvent donc être sur deux serveurs distincts.

Les liens hypertextes sont codés avec la balise `A` et l’attribut `href`. Sur le navigateur, le texte entre début et fin de `A` est souligné pour indiquer qu’il représente un lien vers un autre document. L’attribut `href` est n’importe quelle URL.

Enfin, la balise `body` a un attribut qui définit la couleur de fond de la page. Les couleurs sont notées au format RGB, c’est-à-dire avec une combinaison de rouge, de vert et de bleu. Ces trois couleurs prennent une valeur entre 0 (absence) et 255 (forte présence) pour que leur combinaison donne la couleur désirée. Les valeurs sont données ici en base 16 (en hexadécimal) dont la représentation utilise les chiffres de 0 à 9 et les lettres de A à F. Ici le code `"#FFFFFF"` signifie : le rouge, le bleu et le vert au maximum, ce qui donne du blanc.

4.3 Images

Tous les formats d'image ne sont pas affichables par votre navigateur. Le plus souvent, sans configuration particulière, il manipule sans difficultés les images au format GIF de compuserve (extension `.gif`) ou JPEG (extension `.jpg`) ou Portable Network Graphics du M.I.T. (extension `.png`).

Le format d'enregistrement le plus simple des images, le bitmap, est aussi le plus coûteux en mémoire. Sur le réseau où les temps de transfert sont critiques, ce format est inutilisable. Les formats cités précédemment, réalisent une compression de l'information qui permet de réduire énormément la taille des fichiers (et donc les temps de transfert) sans porter ou trop porter atteinte à la qualité de l'image.

Le format gif n'utilise que 256 couleurs, alors que le jpg ou le png en autorise beaucoup plus. Le format jpg réalise une compression avec perte d'information, c'est-à-dire une baisse de la qualité de l'image, mais cette dégradation n'est pas ou peu visible.

4.4 Formulaires

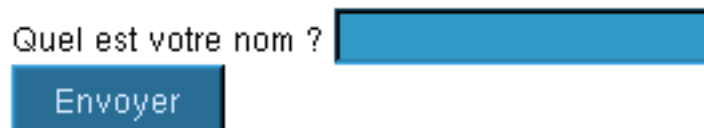
Un formulaire est un écran de saisie dans la fenêtre du navigateur. Pour construire un formulaire, seul le langage HTML suffit. Par contre, le traiter nécessite l'intervention d'un programme résidant sur un serveur Web. Une fois constitué, le formulaire permet à l'utilisateur de saisir des données, qui seront envoyées au serveur et traitées par le programme associé. Nous vous reportons à la section 5 pour étudier la façon de traiter les données sur le serveur, et le moyen d'écrire ces programmes. Cette partie n'est dédiée qu'à l'étude des balises mises à votre disposition pour construire le masque de saisie.

Commençons par un exemple simple :

```
<html>
<head>
  <title></title>
</head>
<body>
  <form action="http://serveur/programme" method="get">
    Quel est votre nom ? <input type="text" name="nom"><BR>
    <input type="submit" value="Envoyer">

  </form>
</body>
</html>
```

et le rendu dans le navigateur :



Dans la zone face au message *Quel est votre nom ?*, l'utilisateur a la possibilité de saisir un texte et l'appui du bouton *Envoyer* enverra ce texte au programme dont l'URL apparaît comme valeur de l'attribut `action` de la balise `form`.

Voici un exemple plus complet des possibilités des formulaires :

```
<html>
<head>
  <title></title>
```

```

</head>
<body bgcolor="white">
  <form action="http://serveur/programme" method="get">
    <b>Quel est votre nom ?</b>
    <div align="center">
      <input type="text" name="nom">
    </div>
    <b>Quel est votre sexe ?</b>
    <div align="center">
      Fille : <input type="radio" name="sexe" value="G">
      Garçon : <input type="radio" name="sexe" value="F">
    </div>
    <b>Quels sont vos loisirs préférés :</b>
    <div align="center">
      <select name="loisirs">
        <option>l'informatique
        <option>le bateau
        <option>le sport
        <option>le cinéma
      </select>
      <select multiple name="autresfacon">
        <option> l'informatique
        <option>le bateau<option>le sport
        <option>le cinéma
      </select>
    </div>
    <b>Avez-vous</b>
    <div align="center">
      un vélo ? <input type="checkbox" name="velo">,
      une voiture ? <input type="checkbox" name="voiture">
    </div>
    <b>Voulez-vous ajouter quelques commentaires ?</b>
    <div align="center">
      <textarea name="comment">
Placez vos commentaires ici
      </textarea>
    </div>

    <input type="submit" value="Envoyer">
    <input type="reset" value="Annuler">

  </form>
</body>
</html>

```

et le rendu dans le navigateur :



4.5 Feuilles de style

Les *feuilles de style* (*CSS* ou *Cascading Style Sheets*) sont un moyen original pour contourner un premier déficit de la première norme HTML, à savoir l'imbrication d'éléments de présentation avec le reste du texte. Grâce aux CSS, il est possible de décrire, soit en entête d'un document, soit dans un autre document ce qu'est la mise en forme d'un élément structurel.

Par exemple, si nous voulons que tous les titres de niveau 1 soient centrés et en couleur rouge, nous sommes obligés d'écrire

```
<H1 ALIGN="center"><FONT color="red">Voici un titre</FONT></H1>
```

et ceci pour chaque titre de niveau 1. Si le document est grand ou si le site est composé de nombreux documents qui doivent être de présentation uniforme, c'est très pénalisant.

Utiliser les CSS permet de décrire un style et donc d'exprimer que tous les titres de niveau 1 par exemple sont centrés et en couleur rouge à l'aide d'une seule commande :

```
h1 {
color: red;
text-align: center
}
```

Depuis l'apparition des feuilles de style, presque tous les attributs de mise en forme des balises HTML sont désuètes (*deprecated* en anglais), mais elle fournissent plus d'éléments de présentation que le HTML de base n'en fournissait. Par exemple, des tailles de marge droite et gauche.

```
p {margin-left: 5cm;
margin-right: 5cm;
color: blue
}
```

Cet exemple définit un style qui met tous mes paragraphes en bleu et avec des marges droite et gauche de 5cm. Mais alors, comment faire si j'ai deux (ou plus) types de paragraphes dans mon document ? Je voudrais des paragraphes pour y mettre des citations en exergue et d'autres pour y mettre des informations normales. Voici un exemple plus complet.

```
<html>
<head>
  <title>ess</title>
  <style TYPE="text/css" MEDIA=screen>
    <!--
    h1 { color: red; text-align: center }
    P{ color: black; background: transparent}
    .citation{margin-left: 5cm; margin-right: 5cm;
              background: yellow; color: blue }
    .auteur{ align: right; color: green}
    -->
  </style>
</head>
<body bgcolor="white">
<h1> Des titres, citations et auteurs </h1>
<div class=citation> Il vaut mieux qu'il pleuve un jour comme aujourd'hui,
  qu'un jour ou il fait beau!
</div>
<div class=auteur>Pierre Dac</div>
<p>Est-ce convaincant ? </p>
</body>
</html>
```

Les balises de commentaires <!-- et --> sont juste là pour faire en sorte que les navigateurs qui ne prennent pas en charge (on dit aussi *ne supportent pas*) les feuilles de style ne tiennent pas compte de ces instructions.

Chapitre 5

Programmation sous Internet

5.1 Programmation sur un serveur WEB

Nous avons étudié comment s'établissait le dialogue entre un client et un serveur dans le cas d'Internet et particulièrement dans le cas du service HTTP de communication de pages hypertexte (cf 2.4). Offrir la possibilité de programmer sur le serveur Web est offrir la possibilité que la page Web consultée par le client ne soit pas un objet construit à l'avance mais à la demande. Le Web devient alors dynamique dans son contenu. L'application la plus connue de ce principe est le moteur de recherche (cf 2.6). La question d'un internaute génère la construction d'une page sur mesure et pour cela, un programme a été utilisé.

5.1.1 Pourquoi ?

La motivation principale est d'avoir des pages Internet dont le contenu soit *dynamique*. Un problème important pour les administrateurs de sites WEB est la façon dont doit évoluer le site. Trop souvent, les sites contiennent des informations anciennes, dépassées. Comment opérer pour être sûr de fournir toujours des informations cohérentes, une présentation homogène ? La réponse est classique : utiliser des procédures automatiques et une source de données unique. La programmation et les bases de données sont une réponse informatique traditionnelle.

Nous devons donc définir un modèle de fonctionnement qui permette de varier le contenu d'une page en fonction des interactions avec l'utilisateur, du contenu d'une base de données. De nombreuses solutions existent déjà. Examinons les maintenant.

5.1.2 Solutions

CGI

La première solution imaginée par les concepteurs du WEB est la passerelle **CGI**. Le fonctionnement est simple : une URL désigne non plus un document mais un programme sur le serveur. La passerelle CGI est une norme qui dit comment l'URL doit être construite, en particulier comment passer des données en entrée du programme, et comment le programme doit fonctionner.

Dans la norme CGI, une URL est de la forme :

```
http://serveur/dossier/programme?argument1=valeur1&argument2=valeur2
```

Cet exemple signifie que le client qui envoie cette requête au **serveur** demande l'exécution de **programme** dans le dossier **dossier** avec en entrée deux données identifiées par les noms **argument1** et **argument2** et qui prennent pour valeurs respectivement **valeur1** et **valeur2**.

La réponse envoyée par le serveur est le résultat d'exécution du programme en question. Il est donc impératif que ce programme retourne un document HTML ou encore un type MIME.

Un inconvénient de la passerelle CGI est sa difficulté de mise en oeuvre pour les utilisateurs novices. Beaucoup de choses assez compliquées doivent être prises en charge par le programme à

commencer par la lecture des arguments en entrée, qui n'est pas simple. C'est souvent du ressort d'un informaticien de se charger de la création de tels programmes. Par contre, un avantage certain est la souplesse. Les programmes peuvent être écrits dans tout langage interprété ou compilé. Les performances peuvent donc être de qualité.

La plupart des moteurs de recherche fonctionnent à l'aide de passerelles CGI. Le langage le plus utilisé est certainement le langage perl, très efficace pour la manipulation de textes.

D'un point de vue technique, l'exécution du code CGI n'est pas réalisé par le serveur Web, mais dans une tâche (ou processus) indépendant. La norme CGI exprime les conditions dans lesquelles les deux tâches que sont le serveur Web et l'exécution du code CGI dialoguent. C'est la grande différence avec la technique que nous présentons dans la section suivante. On peut présenter le dialogue qui s'établit entre client et serveur comme dans la figure 5.1.

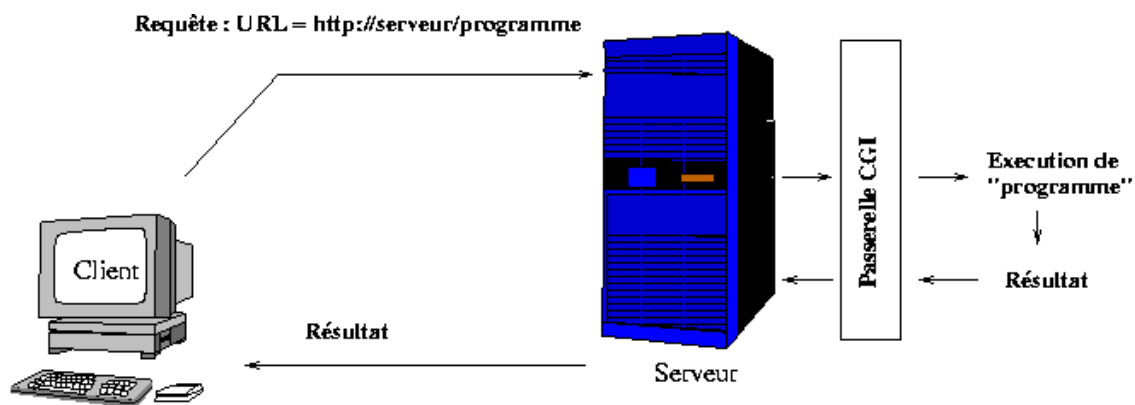


FIG. 5.1 – Dialogue client/serveur dans le cas de l'exécution de programmes CGI.

Code dans le HTML

Voici une solution nouvelle et très intéressante pour les utilisateurs qui ne sont pas forcément informaticiens mais qui ont une connaissance technique minimale. Écrire du code directement dans la page HTML est très simple. Le principe de fonctionnement est le suivant.

Le client émet une requête comme à l'habitude. Le document demandé contient du code dans un langage adapté. Le serveur commence par exécuter ces parties de code dont le résultat sera inséré dans la page HTML en question.

Nous allons explorer un langage particulier : PHP. Les alternatives sont

- Perl encore, très agréable et très puissant mais peut-être un peu plus difficile à apprendre.
- Les SSI ou *server side include*. Assez simples mais peu fonctionnels par rapport à PHP. Elles ont fait l'objet de normalisations et c'est là un avantage certain.
- ASP de Microsoft, beaucoup moins riche et performant, ne fonctionnant que pour un seul type d'architecture et de serveur WEB.
- Beaucoup d'autres solutions *propriétaires* souvent fournies avec des systèmes de gestion de bases de données. Le gros inconvénient est justement que ces solutions sont propriétaires, c'est-à-dire conçues pour une seule base de données, d'un seul constructeur ou éditeur.

5.1.3 Introduction à PHP

People Hate Perl, *Personnal HomePages*, ou même une définition réursive *PHP : Hypertext Preprocessor*, les significations choisies pour cet acronyme ne manquent pas. PHP est un langage de programmation dédié à Internet, directement inclus dans les pages WEB. On appelle aussi ce genre de langage, un langage de *script*.

Les avantages de PHP sont sa facilité d'utilisation, sa très grande richesse fonctionnelle notamment vis à vis de la connexion à des bases de données. PHP est disponible pour énormément de plate-formes, compatible avec beaucoup de serveurs WEB. Son utilisation se généralise très rapidement. On peut parier que PHP est en train de devenir un incontournable du WEB. Enfin, PHP est un produit à *code source ouvert*¹, ce qui lui offre beaucoup de garanties de sécurité, de mise à jour rapide, et aussi la gratuité.

Un inconvénient pourrait être sa lenteur en comparaison avec des programmes compilés écrits en C accessibles via la passerelle CGI, mais en général, PHP se révèle comme un langage fournissant des exécutions assez rapides.

Un premier exemple de page écrite avec PHP est celle donnée dans le manuel officiel :

```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<?php echo "Salut, je suis un script PHP!"; ?>
</body>
</html>
```

Le serveur WEB qui contient cette page va procéder de la façon suivante pour délivrer ce document.

1. Identifier que le document indiqué est bien du PHP (en général grâce à l'extension du fichier (.php3)).
2. Rechercher à l'intérieur toutes les parties comprises entre <?php et ?>. D'autres balises peuvent toutefois être utilisées.
3. Interpréter les parties identifiées à l'étape précédente. Cette étape fournit un résultat qui remplacera les parties entre <?php et ?>.
4. Retourner enfin au client la page ainsi construite.

5.2 Côté client

Rendre les pages plus attrayantes et effectuer quelques petites tâches sont des actions qui peuvent être prises en charge par des langages de programmation à destination des clients WEB (*i.e.* les navigateurs). Nous recensons maintenant les techniques les plus courantes.

La bataille est rude dans ce domaine de l'Internet et de nombreuses compagnies s'affrontent par le biais de techniques propriétaires. En effet, les langages et protocoles du WEB sont le plus souvent normalisés. Pour les géants des interfaces WEB client que sont Netscape et Microsoft, la tentation est grande d'enrichir les fonctionnalités de leur navigateur avec des techniques propriétaires pour tenter de fidéliser, attirer et finalement enfermer l'internaute. Les extensions du HTML, comme les frames par exemple, étaient la première initiative de Netscape dans cet esprit.

Le principe de ces extensions est de permettre aux clients de prendre en charge quelques traitements. Certains sont utiles, beaucoup sont à ranger dans le rayon des gadgets. Nous présentons par la suite les techniques les plus répandues : les plugins, les applets, les scripts qui tous les trois se déclinent en de nombreuses versions, variations. La nature de ce que doit faire le navigateur en présence de tel ou tel mécanisme peut évoluer. Donc, sa réaction doit être modifiable (paramétrable) par l'utilisateur.

Pour comprendre ce paramétrage, une façon est de regarder quelles actions il associe à quel type de document. Les types des documents sont identifiés par deux moyens : le premier le plus sûr est son type MIME (cf 3.6.1), le second est l'extension du fichier qui contient le document. Toutes ces associations sont configurables sur votre navigateur. Pour Netscape par exemple, cette configuration est accessible par le menu *Édition-préférences-navigateur-applications*.

¹C'est-à-dire avec une licence similaire à celle de ce document

Une URL désigne un document sur le WEB. Lorsque le navigateur doit l'afficher, il analyse son type, recherche dans sa configuration l'association qui lie l'application à ce type de document. Il effectue ensuite la tâche appropriée.

5.2.1 Plugins

Les plugins sont des extensions des navigateurs. Ce sont des logiciels. Ils sont disponibles le plus souvent gratuitement sur Internet. Ils permettent de

- afficher des animations simples : Macromedia Flash, Shockwave sont les plus connus.
- afficher des films video/audio : RealPlayer par exemple.
- afficher des documents particuliers : acrobatreader pour les fichiers PDF,
- et bien d'autres.

Un plugin est particulier à un navigateur et à une architecture matérielle ou un système d'exploitation. Cette technique va à l'encontre du principe de l'universalité du client.

Dans le cas d'un plugin, l'exécution aura lieu *dans* la fenêtre du navigateur.

5.2.2 Les formulaires

Les formulaires sont des écrans de saisie pour les postes clients. Pour construire un formulaire, seul le langage HTML suffit. Par contre, pour le traiter, cela nécessite l'intervention d'un programme résidant sur le serveur (CGI ou PHP par exemple).

5.2.3 Applets

Les applets sont des programmes écrits pour la machine virtuelle Java. Lorsqu'un document contient la référence à une applet, elle est téléchargée en mémoire, et exécutée sur la machine Java qui est le plus souvent intégrée dans votre navigateur.

Cette solution est à privilégier dans une solution intranet qui supporte mieux de gros transferts. En effet, une applet est souvent volumineuse et donc longue à télécharger. De plus, les machines virtuelles Java sont parfois différentes d'une machine à l'autre, d'un navigateur à un autre. Il en résulte des conflits et des incompatibilités. Finalement, les applets ne sont pas toujours aussi sûres qu'attendu.

5.2.4 JavaScript et VBScript

Ce sont des langages de script dont le code est directement inclus dans les pages web. Le code est téléchargé sur le client qui a la charge de l'exécuter. Nous écartons immédiatement VBScript qui ne fonctionne qu'avec le navigateur de Microsoft. JavaScript peut être utile pour

- Animer des pages. Les exemples les plus connus sont les images qui changent lorsque le curseur de la souris passe dessus.
- Contrôler la validité de la saisie dans les formulaires. On peut par exemple vérifier qu'un âge sera bien compris entre 0 et 150 (!) ou qu'une adresse électronique sera bien de la forme `user@domaine`.

Cette fonctionnalité peut se révéler intéressante pour limiter les échanges entre clients et serveurs.

- Activer des actions lorsqu'une page est chargée, quittée, etc. Ces contrôles sont très utiles par exemple pour fermer une connexion à une base de données.

JavaScript est un langage de programmation dans le style de la syntaxe Java, mais n'a pas grand chose d'autre à voir avec Java. C'est un langage de programmation événementielle.

Chapitre 6

PHP et Javascript

Nous donnons ici un rapide survol des langages PHP et Javascript. Plus d'éléments sont donnés sous forme d'exercices voir <http://www.grappa.univ-lille3.fr/~tommasi/php>

6.1 PHP

6.1.1 Échappement, constructions

Le code PHP est inséré dans du code HTML entre les deux balises `<?php` et `?>` ou `<? et ?>` ou encore `<SCRIPT LANGUAGE="php">` et `</SCRIPT>`.

Il est possible de mêler le code PHP et le code HTML de façon assez surprenante comme dans l'exemple suivant affiche 10 lignes de *Toto s'en va*.

```
<html>
<head>
<title></title>
</head>
<body>
<? for($i=1;$i<=10;$i++) { ?>
Toto s'en va<BR>
<? } ?>
</body>
</html>
```

La syntaxe du langage est similaire à celle du C ou du Java. Les instructions sont terminées par des `;`. Les blocs d'instructions sont entourés par des `{` et `}`.

Les commentaires sont précédés de `//` ou entourés de `/*` et `*/`.

6.1.2 Variables

PHP est aussi un langage orienté objet. On peut utiliser les types de base simples comme les entiers (`integer`), les réels (`double`), les chaînes de caractères (`string`), mais aussi des types plus évolués comme les tableaux associatifs (`array`) et les objets (`object`).

La déclaration de type n'est pas requise et l'interprète se charge de gérer les types seul. C'est suffisant dans bien des cas.

Une variable est syntaxiquement identifiée par le préfixe `$` et son nom ne peut pas comporter d'espaces. Les majuscules et minuscules sont importantes (exemple : `$mvariable`, `$maVariable` sont deux variables différentes.)

Les chaînes de caractère sont entourées de `"` ou de `'`. La deuxième construction demande de considérer tous les caractères dans la chaîne tels qu'ils sont. Dans une chaîne de caractères

entourée de ", les variables sont remplacées par leur valeur et d'autres séquences de caractères ont une signification spéciale. Par exemple, la séquence `\n` signifie *retour à la ligne*.

```
$bag='coucou';  
$nom1='big$bag\n';  
$nom2="big$bag\n";
```

La variable `$nom1` contient exactement les caractères `big$bag\n`. La variable `$nom2` contient `bigcoucou` et se termine par un retour à la ligne (invisible ici, mais qui peut avoir un effet quand on l'imprime à l'écran ou la sauve dans un fichier).

Les expressions dont la valeur est nulle sont interprétées dans une condition comme la valeur **Faux**. Les autres sont considérées comme **vraies**.

6.1.3 Opérateurs

Ils sont identiques à ceux permis dans le langage C et ses dérivés syntaxiques (C++, java, perl,...). Voici les plus utilisés.

Les opérateurs arithmétiques usuels + - = / *.

Les opérateurs booléens et de comparaison ! (non), && (et), || (ou), < (inférieur), > (supérieur), == (égal), >= (supérieur ou égal), <= (inférieur ou égal).

Notez comme d'habitude la différence entre le signe d'affectation = et le signe de comparaison ==. Un opérateur oublié et pourtant très utile dans bien des cas est `cond?exproui:exprnon` qui permet de se passer d'une construction `if ... else ...` avantageusement. Par exemple :

```
$reponse=$abbrev=='0'?'Oui':'Non';  
// identique à :  
if ($abbrev=='0') {  
    $reponse = 'Oui';  
} else {  
    $reponse = 'Non';  
}
```

Enfin, l'opérateur `.` met bout à bout (concatène) deux chaînes de caractères.

6.1.4 Structures de contrôle

Les conditions doivent apparaître entre parenthèses et les blocs d'instructions entre accolades.

```
if (cond) {... } elseif {...} else {...}.  
while (cond) {...}  
do {...} while (cond)  
for(debut;cond;action){...}
```

6.1.5 Fonctions

L'emploi de fonctions facilite la maintenance et la lisibilité du code. De nombreuses fonctions sont disponibles dans la distribution de PHP et aussi sur Internet. Voyons un exemple de définition de fonction, puis son utilisation.

```
function pied($compteur,$nom_auteur='tommasi',$email='tommasi@univ-lille3.fr') {  
    global $lbl_deb_compteur, $lbl_fin_compteur;  
  
    echo '<address><a href="mailto:'. $email. '">'. $nom_auteur. '</a>';  
    if ($compteur!="") {
```

```

        echo "\n".$lbl_deb_compteur.$compteur.$lbl_fin_compteur;
    }
    echo '</address>';
    echo "<br>Dernière modification".date( "F d Y H:i:s.",getlastmod() );
    echo '</html>'. "\n";
}

```

L'exemple définit une fonction dont le nom est `pied` qui affiche le bas d'une page HTML en tenant compte de quelques paramètres. L'objet d'une telle fonction serait d'uniformiser la présentation d'un ensemble de pages. Elle accepte de 1 à 3 paramètres. Le premier est un argument obligatoire. Les deux autres sont optionnels et ont une valeur par défaut indiquée après le signe `=` .

Voici trois appels possibles et leurs correspondance.

```

<? pied('comp1'); // pied('comp1','tommasi','tommasi@univ-lille3.fr') ?>
<? pied('comp1','Marc'); // pied('comp1','Marc','tommasi@univ-lille3.fr') ?>
<? pied('comp1','Jules','jules@coucou.org'); ?>

```

Toute variable déclarée ou utilisée dans le corps d'une fonction est considéré comme locale à la fonction. Cela signifie qu'elle est inconnue à l'extérieur de celle-ci. Inversement, une fonction ne connaît les variables globales (déclarées ou utilisées ailleurs que dans une fonction) que si elles apparaissent dans une déclaration spécifique. C'est l'objet de la deuxième ligne de cet exemple. Les valeurs des variables globales pourraient être :

```

$lbl_deb_compteur='<BR>';

```

La ligne

```

    echo "<br>Dernière modification".date( "F d Y H:i:s.",getlastmod() );

```

illustre la façon selon laquelle les fonction s'imbriquent et s'utilisent. La fonction `date` met en forme une date et la fonction `getlastmod` obtient la date de dernière modification d'un document. Cette ligne a pour objet d'afficher, sous forme lisible, la date de la dernière modification du document consulté.

La fonction `getlastmod` est remplacée par le résultat de son exécution, illisible en général, dans l'appel à `date` . Ensuite, `date` le met en forme lisible. Ce résultat est mis au bout de la chaîne `
Dernière modification` et le tout est affiché par la fonction `echo` .

6.1.6 Tableaux

Les tableaux en PHP sont indicés soit par des entiers soit par des chaînes de caractères. Les indices apparaissent entre crochets. Pour ajouter un élément à un tableau indicé par des entiers, il est inutile de préciser l'indice (voir le troisième exemple ci-dessous).

```

$tab[0]='Hello';
$tab[1]='Salut';
$tab[]='Olé';
$assoc["blanc"]="#FFFFFF";
$assoc["noir"]="#000000";

```

Pour parcourir un tableau, on utilise souvent la structure suivante :

```

while (list($indice,$valeur) = each($tab) ) {
    ...
}

```

La fonction `each` parcourt le tableau `$tab` et retourne deux valeurs, l'indice et la valeur du tableau à cet indice. Ces deux valeurs peuvent être rangées dans deux variables avec une seule construction utilisant la fonction `list` .

6.1.7 Traitement de formulaires

PHP est très adapté au traitement des formulaires HTML. Les formulaires sont des masques de saisie apparaissant dans la fenêtre du navigateur et invitant les utilisateurs (donc sur le client) à saisir des informations. Nous avons vu en section 4.4 comment construire de tels écrans. Les opérations de traitement des données saisies dans ces formulaires sont des programmes exécutés sur le serveur Web. Traditionnellement ce sont des programmes CGI, mais Ils peuvent être aussi écrits en PHP3. C'est très simple. Voici un exemple. Le formulaire est stocké dans un document html nommé `formdoc.html`

```
<html>
<head>
  <title></title>
</head>
<body>
  <form action="pgm.php3" method="get">
    Quel est votre Nom ? <input type="text" name="zone1">
    Quel est votre prénom ? <input type="text" name="zone2">
  </form>
</body>
</html>
```

avec l'action associée `pgm.php3` :

```
<html>
<head>
  <title></title>
</head>
<body>
  <? echo " bonjour $zone1 $zone2" ?> !
</body>
</html>
```

La zone de saisie définie dans le code HTML du formulaire est associée au nom `zone1` avec l'attribut `name`. Le programme associé à l'action du formulaire a alors accès directement à une variable `$zone1` dont la valeur est celle saisie dans le formulaire (cf 6.1).

Plus joli encore, le code de traitement d'un formulaire est souvent contenu dans le même document que le formulaire lui-même. Ici le document listé s'appelle `invite.php3`.

```
<html>
<head>
  <title></title>
</head>
<body>
<?
  function affiche_form () {
  ?>
    <form action="invite.php3" method="get">
    Quel est votre prénom ? <input type="text" name="prenom">
    <input type="submit"></form>
  <? }

  if (isset ($prenom) ) {
    echo "Bonjour $prenom";
  } else {
    affiche_form();
```

```

    }
    ?>
</body>
</html>

```

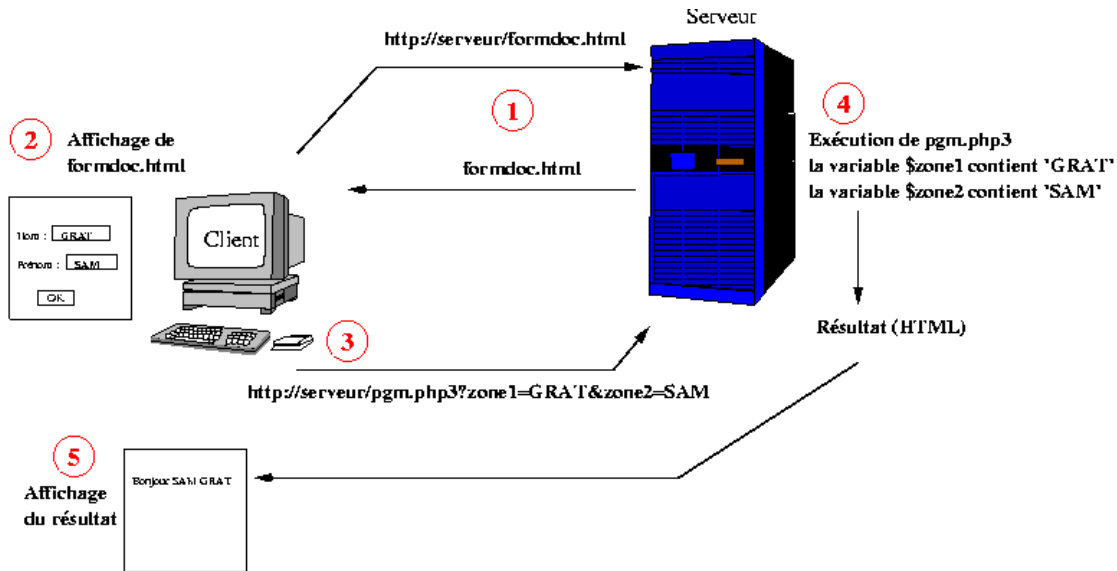


FIG. 6.1 – Traitement d'un formulaire en PHP

6.1.8 Accès aux bases de données

Un point fort de PHP est la facilité avec laquelle on accède à des données mémorisées dans une base de données. PHP permet la connexion aux plus grandes bases du marché. Par exemple nous donnons l'exemple d'un script PHP qui se connecte à une base PostgreSQL, et affiche la liste des noms de clients.

```

<html>
<head>
  <title></title>
</head>
<body>
<?
  $connection=pg_connect("dbname=clients user=tom password=coucou");
  $requete="SELECT nom from clients";
  $result=pg_Exec ($connection,$requete);
  for($i=0; $i<= pg_numrows($result);$i++) {
    $arr = pg_fetch_array ($result, $i);
    echo $arr["nom"]."<BR>";
  }
  ?>
</body>
</html>

```

Nous laissons aux exercices l'examen plus poussé des possibilités d'interaction entre PHP et les bases de données.

6.1.9 Fonctions usuelles, exemples et exercices

Premiers exercices avec des formulaires

1. Écrire un formulaire avec deux zones de saisie pour permettre à un utilisateur de saisir son nom et son prénom. En retour, le programme doit lui dire bonjour.
2. Écrire un formulaire avec une zone de texte de 5 lignes sur 80 caractères. Votre programme devra mettre le texte saisi en majuscules et l'écrire en rouge. Vous pouvez utiliser la fonction `strtoupper` et la fonction `nl2br`.
3. Écrire un formulaire qui pose une question et propose dans une zone de liste déroulante 3 réponses possibles. Le programme devra récompenser l'utilisateur qui trouve la bonne réponse.
4. Écrire un formulaire avec un deux cases d'option donc non exclusives pour demander à l'utilisateur s'il aime le sucre ou le sel. Remarquez que les deux cases doivent avoir des noms différents.
5. Écrire un formulaire avec un deux cases à cocher exclusives pour demander à l'utilisateur si l'est un garçon ou une fille. Remarquez que les deux cases doivent avoir le même pour nom pour être exclusives et une valeur différente pour les distinguer.
6. Écrire un formulaire avec deux zones de saisie. Votre programme devra afficher le résultat de la multiplication de ces deux nombres. Utilisez le même document pour demander les deux nombres et afficher le résultat.
7. Écrire un formulaire avec deux zones de saisie. Votre programme devra afficher le résultat de la multiplication de ces deux nombres. Utilisez le même document pour demander les deux nombres et afficher le résultat. Utilisez la fonction `isset` pour rendre l'affichage plus joli.
8. Écrire un formulaire avec deux zones de saisie. Votre programme devra afficher le plus petit de ces deux nombres.
9. Écrire un formulaire avec deux zones de saisie. Votre programme devra compter de 1 en 1 entre ces deux nombres. NB : Vous pouvez d'abord chercher le plus petit.
10. Écrire un formulaire avec une zone de saisie. Votre programme doit trier une liste de mots séparés par des virgules. Vous utiliserez les fonctions `split` et `count`.

La fonction `isset`

Détermine si une variable est définie, (c'est-à-dire si elle existe). Par exemple, `isset($a)` prend la valeur vrai si `$a` est une variable définie. La fonction `isset` est très souvent utilisée lorsque l'on traite les formulaires

Exemple :

Le code suivant contenu dans un document `quest.php3` affiche un formulaire ou un message bonjour selon le cas. Le formulaire est affiché à la première utilisation du fichier, c'est-à-dire lorsqu'un utilisateur demande pour la première fois le document `quest.php3`. Le message bonjour est affiché lorsque l'utilisateur valide le formulaire.

```
<html><head></head><body>
<? if (isset($nom)) {
    echo "Bonjour $nom";
} else {
    <FORM action="quest.php3" method="get">Votre nom ?
        <input type="text" name="nom"><input type="submit">
    </form>
}</body></html>
```

■

Exercice :

Traiter un formulaire dans un seul document php, qui demande son nom, son prénom, son adresse email. Le prénom est optionnel les autres champs ne le sont pas. Le programme doit tester si tous les champs obligatoires ont été saisis, re-proposer le formulaire avec les valeurs déjà saisies le cas échéant et envoie un message de félicitations quand tout est bon.

La fonction mail

Envoie un message électronique. La fonction accepte trois paramètres, l'adresse, le sujet et le corps du message.

Exemple :

```
mail("tommasi@univ-lille3.fr",  
     "Le sujet du message",  
     "le message en question...");
```

■

Exercice :

Écrire un petit client mail HTTP, à la façon des caramail, hotmail et consors. Vous mettez à disposition des utilisateurs une page avec une zone de saisie pour le nom des destinataires, une autre pour le sujet, une dernière pour le texte. Vous ajoutez un bouton pour l'envoi du message.

La fonction require

Insère un fichier dans un document php. Cette fonction est très utile pour plusieurs tâches : uniformiser la présentation de documents HTML en définissant les entêtes, les pieds de page... ; inclure des fonctions définies par l'utilisateur et souvent utilisées ; inclure des définitions de variables ou constantes globales.

Exemple :

La fonction `pied()` qui apparaît dans le poly.

■

Exercice :

Faire une fonction de conversion de francs en euro et une autre d'euros vers les francs. Le taux de change est à 6,5F pour 1 euro.

Faire une page web qui propose une zone de saisie et deux boutons d'option qui offrent une interface utilisateurs à ces deux fonctions.

Faire une fonction qui affiche un entête de document avec des paramètres qui sont : la couleur de fond, le titre du document.

La fonction die

Arrêter un script en envoyant un message d'erreur. Cette fonction est utile lorsqu'une erreur importante survient.

Exemple :

Tout arrêter si le client est MSIE (on regarde la variable prédéfinie `$HTTP_USER_AGENT` ou on demande l'exécution de la fonction `get_browser` pour obtenir le type de navigateur).

■

La fonction array

Construit un tableau, la structure de données qui permet de stocker des valeurs, ensuite accessibles via une clé ou un indice.

Exemple :

```
$loisirs = ( "Sport" => "Natation", "metier" => "informaticien" ).
On peut aussi faire $loisirs["Sports"] = "Natation";
$loisirs["metier"] = "informaticien";
```

■

La manipulation des tableaux est assez agréable en PHP. Par exemple, on peut utiliser une expression comme `$a[]=10` pour ajouter la valeur 10 au tableau (indiqué par des entiers) `$a`.

Exercice :

Afficher un formulaires qui demande une valeur, et mémorise les 10 premiers nombres impairs suivants dans un tableau. Afficher ensuite le contenu du tableau dans une liste non ordonnée.

Créer un formulaire avec une zone de liste à choix multiple qui contient 10 espèces d'oiseaux (à vous de trouver). Le programme associé doit afficher cette liste. Dans le code HTML du formulaire, vous avez la possibilité de mettre dans l'attribut `name` de la zone de liste **SELECT** un nom de variable terminé par des crochets. L'action pourra récupérer un tableau des éléments sélectionnés.

La fonction count

Compte le nombre d'éléments dans un tableau. Retourne 0 si la variable n'est pas définie, et 1 si la variable est scalaire (n'est pas un tableau).

Exercice :

Améliorer l'exercice précédent pour afficher aussi le nombre d'éléments sélectionnés.

La fonction sort

Trie un tableau. L'ordre choisi est alphabétique. voir `asort`, `ksort`, `rsort`

Exercice :

Améliorer l'exercice précédent pour afficher les oiseaux dans l'ordre alphabétique.

Les fonctions list – each –reset

`list` affecte une liste de variables en 1 seul coup. Cette fonction est très souvent utilisée en combinaison avec la fonction suivante :

`each` retourne la valeur courante de la paire clé/valeur dans un tableau et passe au suivant.

`reset` reprend au début d'un tableau. Le prochain appel de `each` retournera la première paire clé/valeur du tableau.

Exemple :

```
while (list($key, $val) = each($loisirs)) {
    echo "$key => $val<br>"; }
```

produit le résultat :

```
Sport = > Natation metier => informaticien
```

■

Exercice :

Écrire une fonction menu qui affiche un menu en passant en paramètre un tableau indicé par des entiers contenant des tableaux indicés par les champs `intitulé` et `url` qui désignent l'intitulé du menu et l'URL correspondante.

Écrire une application qui propose deux listes, l'une avec des animaux (buse, aigle, pigeon, moineau, corbeau, vautour, goeland...) l'autre avec des habitudes alimentaires (gaines, viande, os, herbe, poissons,...). Le but est d'associer les animaux aux régimes. À chaque tour, l'utilisateur sélectionne un ou plusieurs animaux, un régime. Le programme retire les animaux de la liste, cumule les erreurs. Quand la liste des animaux est vide, le programme affiche le score.

Les fonctions file – readdir – opendir

Ces trois fonctions sont souvent utilisées ensembles. La première permet de lire un fichier complet et de le ranger dans un tableau. Les indices du tableau sont les numéros de lignes. Les valeurs sont le contenu de chaque ligne.

Les fonction opendir et readdir servent à obtenir la liste des fichiers se trouvant dans un répertoire.

Exemple :

```
$rep=opendir('.');
while ($file = readdir($rep)) {
    echo "Le fichier $file est ici !";
}
```

■

Exercice :

On considère que tous les exercices de cette feuille sont contenus dans un dossier appelé Exos. Construire une page Web qui affiche la liste des exercices disponibles avec un lien hypertexte pour chacun d'entre eux.

Ajouter une jolie présentation consistant en un tableau à deux colonnes et une seule ligne. Un exercice apparaît dans la colonne de droite. La liste des exercices sera placée dans la colonne de gauche. Chaque élément de cette liste est précédé d'une boule de couleur jaune (fichier `http :/icons/yellowball.gif`), ou rouge (fichier `http :/icons/yellowball.gif`) pour celui actuellement affiché.

La fonction split

Pour `jj` éclater `ii` une chaîne dans un tableau.

Exemple :

```
$brut="toto titi tata tutu";
$tableau=split(" ",$brut);
```

Alors, `$tableau[0]` contient *toto*, et ainsi de suite...

■

Les fonctions strtoupper – strtolower – ucfirst – nl2br

Peti es fonctions de manipulations de chaîne de caractères. La première met la chaîne en majuscule, la deuxième en minuscules, la troisième met l'initiale en majuscule et enfin, la dernière transforme les retours à la ligne en des instructions `
` du langage HTML.

Exemple :

```
if (strtolower($reponse) == 'oui' ) {
    echo "vous avez donc répondu oui ou Oui ou OUI ou oUI...";
}
```

■

Connexion à une base de données

Nous donnons ici quelques exemples de connexion à base de données PostgreSQL à l'aide de PHP. Le thème est la création d'un outil de communication de brèves sur Internet à la façon SlashDot ou Linuxfr. L'application est très largement simplifiée. Voici les instructions SQL pour la définition des tables :

```

CREATE TABLE themes (numtheme SERIAL PRIMARY KEY,
                      libelle varchar(20) NOT NULL,
                      logo varchar(100));
CREATE TABLE auteurs (numauteur SERIAL PRIMARY KEY,
                      nom varchar(30) NOT NULL, prenom varchar(30),
                      email varchar(50));
CREATE TABLE nouvelles (numnouvelle SERIAL PRIMARY KEY,
                      texte text, reftheme INT4,
                      refauteur int4,
                      FOREIGN KEY (reftheme) REFERENCES themes (numtheme),
                      FOREIGN KEY (refauteur) REFERENCES auteurs (numauteur))

```

La fonction `pg_connect`

Permet de se connecter à un serveur de bases de données postgresql. On doit donner en argument à cette fonction, une chaîne de connexion qui désigne le serveur postgresql, le nom d'utilisateur, son mot de passe et la base de données. En retour, il faut mémoriser l'identificateur de cette connexion qui permettra ensuite d'exécuter des requêtes.

Exemple :

Dans cet exemple, le serveur de base de données est aussi le serveur HTTP et nous supposons que l'utilisateur n'a pas de mot de passe. Si l'identificateur de connexion est nul, alors la connexion n'est pas réussie.

```

$conn_ident = pg_connect("dbname=nouvelles user=tommasi");
if ($conn_ident)
    echo "connexion réussie";
else
    echo "echec de la connexion";

```

■

La fonction `pg_exec`

Permet de lancer une requête SQL au serveur identifié par une connexion. La fonction retourne une référence du résultat.

Exemple :

Nous recherchons les enregistrements de la table thèmes. L'affichage du résultat n'est pas encore effectué.

```

$conn_ident = pg_connect("dbname=nouvelles user=tommasi");
if ($conn_ident)
    echo "connexion réussie";
else
    echo "echec de la connexion";
$req_resu= pg_exec("SELECT * FROM themes");

```

■

La fonction `pg_numrows`, `pg_fetch_array`

Ces deux fonctions permettent de connaître le nombre d'enregistrements dans une référence de résultat de requête et de lire le résultat ligne par ligne.

Exemple :

La fonction `pg_numrows` est utilisée pour construire une boucle *pour* qui parcourt le résultat de la requête. Chaque ligne est lue avec `pg_fetch_array($req_resu, $i)` qui lit la *i*ème ligne

du résultat. Cette instruction retourne un tableau avec une entrée par colonne, accessible par son indice de position ou un indice textuel.

```
$conn_ident = pg_connect("dbname=nouvelles user=tommasi");
if ($conn_ident)
    echo "connexion réussie";
else
    echo "echec de la connexion";
$req_resu= pg_exec("SELECT * FROM themes");
$nombre_enreg = pg_numrows(req_resu);
for($i=0; $i<$nombre_enreg;$i++) {
    $une_ligne= pg_fetch_array($req_resu, $i);
    echo "Numtheme = ".$une_ligne[0];
    echo "libelle= ".$une_ligne[1];
    echo "logo = ".$une_ligne[2];
}
```

On peut accéder au champ libellé par `$une_ligne[1]` ou `une_ligne["libelle"]`.

La fonction `pg_freeresult`

Permet de libérer l'espace mémoire occupé par le résultat d'une requête :

Exemple :

```
$conn_ident = pg_connect("dbname=nouvelles user=tommasi");
if ($conn_ident)
    echo "connexion réussie";
else
    echo "echec de la connexion";
$req_resu= pg_exec("SELECT * FROM themes");
$nombre_enreg = pg_numrows(req_resu);
for($i=0; $i<$nombre_enreg;$i++) {
    $une_ligne= pg_fetch_array($req_resu, $i);
    echo "Numtheme = ".$une_ligne[0];
    echo "libelle= ".$une_ligne[1];
    echo "logo = ".$une_ligne[2];
}
pg_freeresult($req_resu);
```

Bibliographie

- [1] I. Cohen. *CGI/Perl et JavaScript*. Eyrolles, 1996. Bonne référence.
- [2] D. Flanagan. *JavaScript*. Précis et concis. O'Reilly, 2000. Comme souvent dans cette série : précis et concis.
- [3] E. Harold. *Programmation réseau avec Java*. O'Reilly, 1997. Une bonne introduction aux réseaux dans les premières sections.
- [4] L. Lacroix, N. Leprince, C. Boggero, and C. Lauer. *Programmation WEB avec PHP*. Eyrolles, 2000. Avis partagé. De nombreux exemples et une étude de cas.
- [5] R. Lerdorf. *PHP*. Précis et concis. O'Reilly, 2000. Comme souvent dans cette série : précis et concis.
- [6] D. Lowe. *Les réseaux pour les nuls*. Sybex, 1996. Introductif. Intéressant bien que le style soit agaçant.
- [7] R. Orfali, D. Harkey, and J. Edwards. *Client/Serveur, Guide de survie*. International Thomson Publishing, 1995. De nouvelles éditions sont parues depuis pour ce guide de référence un peu technique sur le modèle client serveur.
- [8] A. Tanenbaum. *Réseaux*. Dunod, 1997. Très complet. Très précis. C'est une référence.

Index

A		
attachement	22	
B		
Base64	22	
C		
cache	17	
Cascading Style Sheets	31	
CGI	33	
CHAP	14	
chat	23	
client de messagerie électronique	19	
clients	2	
cookies	17	
CSS	31	
F		
feuilles de style	31	
firewall	17	
G		
gateway	12	
I		
icq	23	
IMAP	18	
indexation	15	
instant messenger	23	
IP	12	
IP masquerade	13	
IPV6	13	
Irc	23	
M		
masque	12	
MIME	13, 22	
modèle en couches	12	
moteur de recherche	14	
P		
PAP	14	
pare-feu	17	
passerelle	12	
POP	18	
port	12	
Q		
quoted-printable	22	
S		
serveur mandataire	17	
serveurs	2	
SMTP	18	
T		
talk	23	
TCP	12	
U		
UDP	12	
PPP		14
proxy		17