Communication client-serveur

d'après un article de David Roche

Deux ordinateurs en réseau peuvent s'échanger des données. Dans la plupart des cas, ces échanges ne sont pas "symétriques" : en effet un ordinateur A va souvent se contenter de demander des ressources (fichiers contenant du texte, photos, vidéos, sons...) à un ordinateur B. L'ordinateur B va se contenter de fournir des ressources à tous les ordinateurs qui lui en feront la demande. On dira alors que l'ordinateur A (celui qui demande des ressources) est un client alors que l'ordinateur B (celui qui fournit les ressources) sera qualifié de serveur.

En tapant «http://www.google.fr» dans un navigateur comme Firefox, votre machine va chercher à entrer en communication avec le serveur portant le nom «www.google.fr» (en fait, la communication va être établie avec le serveur www du domaine google.fr). Une fois la liaison établie, le client et le serveur vont échanger des informations en dialoguant :

- client : bonjour www.google.fr, merci de m'envoyer le fichier `index.html`
- serveur : OK, voici le fichier `index.html`
- client : je constate que des images sont utilisés, merci de me les envoyer
- serveur : OK, les voici

Évidemment ce dialogue est très imagé, mais il porte tout de même une part de « vérité ».

Sur internet, ce modèle **client/serveur** domine assez largement, même s'il existe des cas où un ordinateur pourra jouer tour à tour le rôle de client et le rôle de serveur. Très souvent, des ordinateurs (les clients) passeront leur temps à demander des ressources à d'autres ordinateurs (les serveurs). Par exemple, comme expliqué dans l'exemple ci-dessus on retrouve cet échange **client/serveur** à chaque fois que l'on visite une page web. Il y a de fortes chances pour que votre ordinateur personnel joue quasi exclusivement le rôle de client (sauf si vous êtes un adepte du "**peer to peer**").

N'importe quel type d'ordinateur peut jouer le rôle de serveur, mais dans le monde professionnel les serveurs sont des machines spécialisées conçues pour fonctionner 24h sur 24h. Ils peuvent aussi avoir une grosse capacité de stockage afin de stocker un grand nombre de ressources (vidéos, sons,...).

serveur

Afin d'assurer une continuité de service, dans les sociétés, plusieurs serveurs assurent exactement le même rôle (on parle de redondance). Vous vous doutez bien que Google ne possède pas qu'un seul serveur. En effet, chaque seconde, c'est environ 65000 clients (en moyenne) qui se connectent aux serveurs du moteur de recherche de Google. Aucun serveur, même extrêmement performant, ne serait capable de répondre à toutes ces requêtes. Google, Amazon ou encore Facebook possèdent un très grand nombre de serveurs afin de pouvoir satisfaire les demandes des utilisateurs en permanence. Ces entreprises possèdent des bâtiments entiers appelés « **datacenter** » avec d'immenses salles contenant chacune des centaines ou des milliers de serveurs (ces serveurs sont rangés dans des armoires appelées "**baie serveur**").

salle serveur

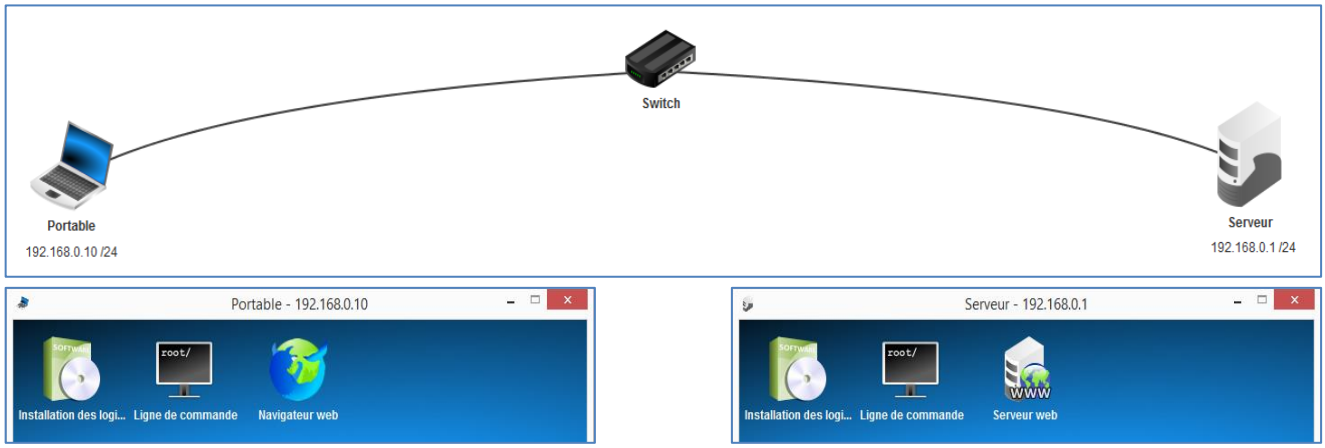
Souvent les serveurs sont spécialisés dans certaines tâches, par exemple, les serveurs qui envoient aux clients des pages au format HTML sont appelés "serveur web".

Il y a quelques années, le web était dit « statique » : le concepteur de site web écrivait son code HTML et ce code était simplement envoyé par le serveur web au client. Les personnes qui consultaient le site avaient toutes accès à la même page, le web était purement « consultatif ».

Les choses ont ensuite évolué : les serveurs sont aujourd'hui capables de générer eux-mêmes du code HTML. Les résultats qui s'afficheront à l'écran dépendront donc des demandes effectuées par l'utilisateur du site : le web est devenu dynamique.

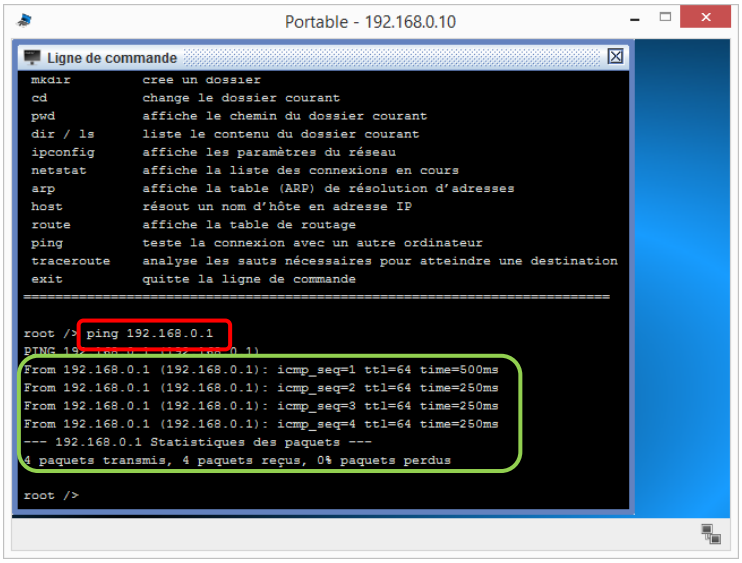
Différents langages de programmation peuvent être utilisés « côté serveur » afin de permettre au serveur de générer lui-même le code HTML à envoyer. Le plus utilisé encore aujourd'hui se nomme PHP. D'autres langages sont utilisables côté serveur (pour permettre la génération dynamique de code HTML) comme Java ou Python. A noter qu'il existe aussi des langages utilisables uniquement côté client comme javaScript (à ne pas confondre avec Java).

Pour illustrer une communication client/serveur simple, on considère la maquette suivante :



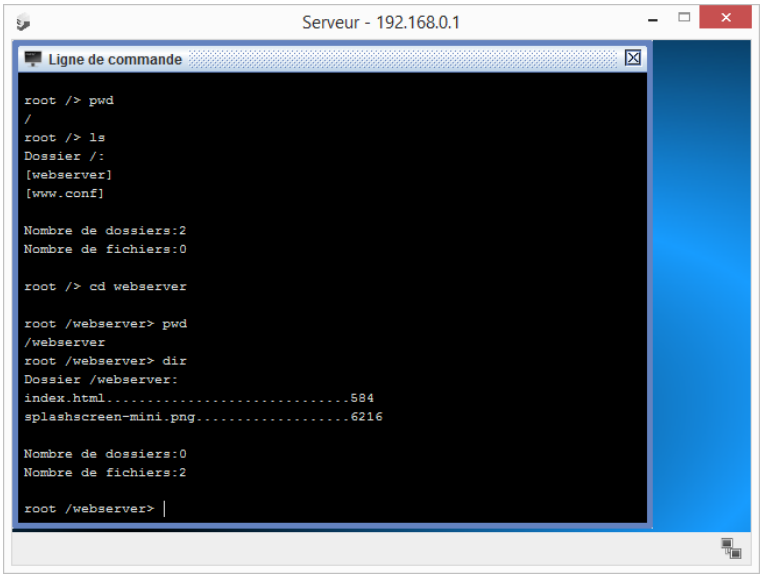
Exercice

1. Avec le logiciel **Filius**, créer cette maquette en attribuant les adresses IP et en installant les logiciels indiqués sur le portable et le serveur.
2. Avec la ligne de commande du portable, tester la connectivité vers le serveur avec la commande ping :



➔ Vous devez obtenir un résultat analogue à celui encadré en vert.
Si ce n'est pas le cas, demander de l'aide avant de passer à la suite.

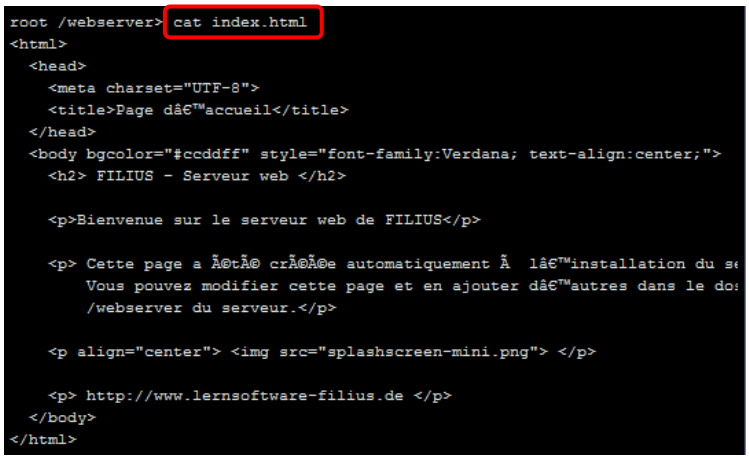
3. Avec la ligne de commande du serveur, tester les commandes **pwd**, **ls**, **cd**, **dir**, comme indiqué dans l'écran suivant :



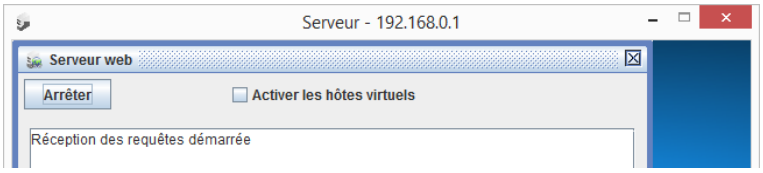
Expliquer le rôle de chacune de ces commandes :

pwd	
ls	
cd	
dir	

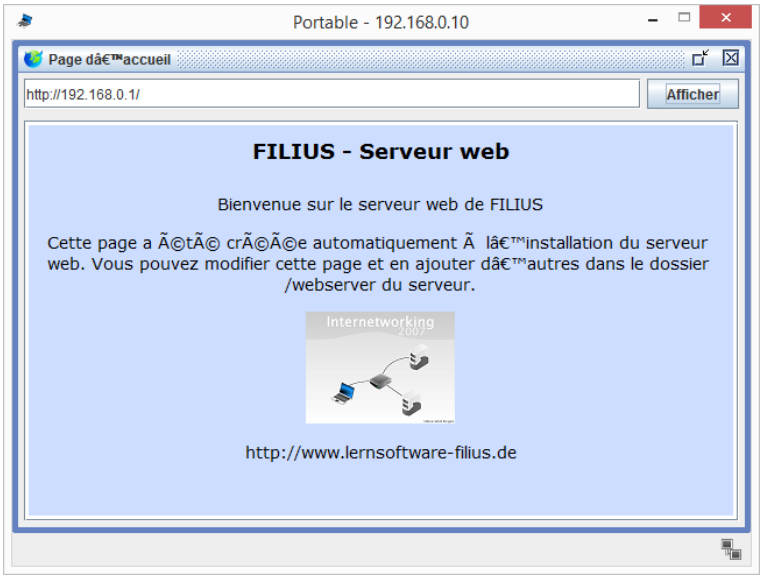
4. Suite à l'étape 3, on constate la présence du fichier **index.html** dans le dossier **webserver**. Visualiser son contenu grâce à la commande **cat** :



Il s'agit du fichier dont le contenu est envoyé par le serveur Web à tout client qui en fait la demande, à condition que le service soit démarré :



5. A partir du navigateur web du portable, tester l'accès à la page **index.html** du serveur :



Pour l'instant, ne pas prêter attention à l'affichage des caractères accentués, mais considérer la petite image affichée dans la page  et répondre aux questions suivantes :

Où est stockée cette image ?	
Quel est son nom ?	
Quelle est sa taille ?	

Défi

Pour terminer, un défi à relever : il s'agit de modifier la page renvoyée par le serveur comme indiqué ci-dessous :