Systèmes informatiques

Franck Guingne,
sur la base du cours d'Olivier Lecarme

Cours Licence 1; Semestre 2

2009–2010

Septième cours : connexion et communication

Plan en cours

1. Connexion à une machine distante
   - Mécanismes élémentaires
   - Mécanismes sécurisés
2. Utilisation de fichiers distants
   - Méthodes simples
   - Méthodes perfectionnées
3. Communication hypertexte
   - Le protocole HTTP
   - Le langage HTML
   - Les navigateurs

Connexion à une machine distante

- la session d'un utilisateur de Unix n'est pas nécessairement ouverte sur le poste de travail depuis lequel il la lance
- la *connexion à distance* permet d'ouvrir une session sur une machine distante, dès l'instant qu'on y dispose d'un compte
- les premiers mécanismes supposaient une liaison directe entre le poste de travail et la machine distante (ligne téléphonique)
- l'existence d'Internet et de réseaux de communication accessibles depuis partout supprime cette restriction
- l'un des problèmes majeurs à résoudre est celui de la sécurité

La commande Telnet

- la commande `telnet` est un outil général de connexion à distance :
  - s'appuie sur les protocoles TCP et IP
  - utilise son propre protocole
  - sert de support à la plupart des protocoles de plus haut niveau (SMTP, HTTP, NNTP, etc.)
- utilisation :
  - `telnet` *`nom de machine`*
  - si la machine accepte le protocole, elle répond comme lors d'une connexion par interface textuelle
  - après identification, connexion distante identique à une connexion locale
  - mode de travail purement textuel
  - fonctionne aussi avec d'autres systèmes que Unix

La commande Telnet (suite)

- le *caractère d'échappement* `C-]` suspend la connexion :
  - passage dans un mode de dialogue interactif
  - invite `telnet>`
  - ensemble de commandes possibles, par exemple :
    - `help` *commande* demande de l'aide sur une commande
    - `open` *nom de machine* établit une connexion
    - `close` termine la connexion en cours
    - `quit` termine la connexion et Telnet
    - `status` indique l'état courant
    - etc.
  - existence de variables affichables et modifiables

Telnet dans Emacs

- dans Emacs, `M-x telnet` ouvre une session
- l'historique est conservé dans le tampon
- des commandes spécialisées :
  - simplifient la saisie de commandes
  - simplifient la modification du texte de la session
  - permettent d'envoyer les commandes propres à Telnet
  - trois menus spécialisés
  - etc.
- utile dès qu'on veut gérer facilement une session de ce type

La commande Rlogin

- la commande `telnet` permet la connexion à tout système qui l'accepte
- la commande `rlogin` ne le permet qu'à un système Unix
- utilisation :
  - `rlogin` *`machine`* `-l` *`nom d'utilisateur`*
  - nom d'utilisateur omis si c'est le même que sur la machine de départ
  - le shell est interactif sur la machine distante
  - la fin de ce shell termine la connexion
  - une ligne ne contenant que `~.` la termine localement
- le mode de travail est normalement purement textuel

Connexion sans mot de passe

- l'identification nécessite le mot de passe
- celui-ci est transmis sans cryptage sur le réseau
- on peut éviter le mécanisme d'identification :
  - placer sur la machine distante un fichier `$HOME/.rhosts`
  - les lignes sont de la forme
    `machine utilisateur`
  - le fichier doit être lisible uniquement par l'utilisateur
  - la connexion est alors acceptée sans identification

Questions de sécurité

- les connexions ordinaires entre machines sont non cryptées
- la transmission par diffusion permet d'intercepter des paquets assez facilement
- la connexion par Telnet est donc très générale mais sans aucune sécurité
- on l'utilise surtout pour des protocoles placés par-dessus :
  - FTP (vu plus tard) sur le port 21
  - Telnet (connexion à distance) sur le port 23
  - SMTP (courrier) sur le port 25
  - DNS (serveur de noms) sur le port 53
  - HTTP (pages Web) sur le port 80
  - POP3 (serveur de courrier) sur le port 110
  - NNTP (groupes de discussion) sur le port 119
  - NTP (service de l'heure) sur le port 123
  - IRC (bavardage) sur le port 194
  - etc.
- pour la plupart de ces protocoles, la sécurité est considérée comme peu importante

Sécurité et protocole de Rlogin

- la connexion à distance est beaucoup plus dangereuse que les protocoles précédents
- une fois la connexion établie, l'utilisateur est beaucoup plus libre qu'avec tout autre protocole
- sans fichier `.rhosts` il faut transmettre le mot de passe sous forme non cryptée
- avec un fichier `.rhosts` on peut se connecter sans mot de passe
- dès qu'on est parvenu à se connecter sur une des machines, on a accès à toutes les autres où un fichier `.rhosts` le permet pour cet utilisateur
- la deuxième solution n'est donc acceptable qu'à l'intérieur d'un réseau local
- le protocole de Rlogin est souvent refusé par les administrateurs
- pour des connexions à travers les réseaux, on utilisera des mécanismes sécurisés

La commande Rsh

- la commande `rsh` permet de faire exécuter une commande à distance
- il n'y a donc pas de processus de connexion
- utilisation :
  - `rsh` *`machine commande`*
  - on peut ajouter l'option `-l` *`utilisateur`*
  - sur certains systèmes il peut y avoir demande de mot de passe
  - le fichier `.rhosts` est utilisé comme pour `rlogin`
  - en fait, appeler `rsh` sans préciser de commande revient à appeler `rlogin`
  - si la commande doit être lancée en travail d'arrière-plan, il faut utiliser l'option `-n` pour éviter un blocage en lecture sur le terminal
- l'insécurité est la même qu'avec Rlogin

La commande Ssh

- la commande `ssh` fait le même travail que Rlogin mais de manière sécurisée :
  - *authentification* du client par le serveur
  - *cryptage* de la transmission
  - garantie d'*intégrité* de cette transmission
- l'utilisation reste très simple
- le *transfert de port* élargit les possibilités
- on peut aussi utiliser le *contrôle d'accès* pour limiter les actions possibles sur le serveur

Utilisation de Ssh

- l'appel se fait comme pour Rlogin :
  - `ssh -l utilisateur machine`
    ou bien
    `ssh utilisateur@machine`
  - en situation normale, simple demande de mot de passe sur la machine distante
  - s'il est correct, établissement de la session
- différences avec Rlogin :
  - l'identité des deux machines reliées est authentifiée
  - le mot de passe est transmis de manière cryptée
  - l'ensemble de la session est également crypté
  - on vérifie l'intégrité des données transmises, c'est-à-dire qu'elles n'ont pas été modifiées sur le trajet

Fonctionnement de Ssh

- fonctionnement sur le modèle client-serveur
- l'authentification du client par le serveur et du serveur par le client se fait par une *clé cryptée* propre à chaque machine
- ces *clés d'hôte* sont conservées après réception (répertoire `$HOME/.ssh`) et permettent de reconnaître un hôte connu
- à la première connexion, l'utilisateur doit approuver ou non l'identité annoncée par le serveur
- cette démarche ne doit être répétée que si le serveur change de clé d'hôte
- on peut remplacer le mécanisme de transmission de mot de passe par un mécanisme plus sûr, la signature par clé publique

Plan en cours

1 Connexion à une machine distante
- Mécanismes élémentaires
- Mécanismes sécurisés

2 Utilisation de fichiers distants
- Méthodes simples
- Méthodes perfectionnées

3 Communication hypertexte
- Le protocole HTTP
- Le langage HTML
- Les navigateurs

Accès direct à des fichiers

- les fichiers locaux sont situés physiquement sur un dispositif physique ou *volume* :
  - partition de disques durs
  - disquette
  - disque compact
  - clé USB
- noter qu'une unité de disques est normalement découpée en partitions

Montage et démontage

- pour qu'un dispositif soit accessible, il doit être *monté* sur un répertoire :
  `mount` *`options volume répertoire`*
- `mount` sans paramètre affiche tous les montages
- l'opération symétrique est le *démontage* :
  `umount` *`répertoire`*
- opérations réservées à l'administrateur pour les volumes fixes
- la plupart des volumes locaux sont montés automatiquement pendant le démarrage de Unix, sur des sous-répertoires du répertoire racine : `/var`, `/tmp`, `/usr`, `/home`, etc.

Montages de volumes non locaux

- le mécanisme *NFS* (*Network File System*) permet de monter un volume non local :
  - l'ordinateur distant doit *exporter* le volume
  - il fonctionne comme serveur NFS
  - le client demande le montage sur un répertoire local
  - c'est souvent fait seulement à la demande (*auto-montage*)
  - la transmission des données est faite selon le protocole UDP, plus simple et rapide que TCP, mais moins sûr (possibilité de pertes d'informations)
- ce mécanisme de serveurs de fichiers permet le partage des données sur un grand nombre de postes de travail
- la commande `df` donne des informations sur l'ensemble des volumes

Copie de fichiers distants

- à travers les montages NFS on a accès à tous les fichiers comme s'ils étaient locaux
- la commande `rcp` permet de sortir du domaine NFS :
  - utilisation des mécanismes de la commande `rlogin`
  - nécessité du fichier `.rhosts` sur les machines distantes
  - forme de la commande :
    `rcp [options] source source ... destination`
  - la référence à un fichier peut être :
    - locale, avec la forme ordinaire
    - distante, de la forme :
      `[utilisateur]@machine:[chemin d'accès]`
      - si l'utilisateur est le même on peut l'omettre
  - exemple : `rcp .ssh/id_dsa.pub guingne@sesame-mips.unice.fr:.ssh/copieclef`

Copie sécurisée

- la commande `scp` utilise le protocole de Ssh
- forme d'appel identique à celle de `rcp`
- la commande demande les mots de passe si nécessaire
- la sécurité est celle de Ssh
- on peut donc copier des fichiers entre deux machines distantes
- exemple : `scp .ssh/id_dsa.pub guingne@sesame-mips.unice.fr:.ssh/copieclef`

Transfert de fichiers par FTP

- le protocole *FTP* (*File Transfer Protocol*) :
  - s'appuie sur les protocoles TCP, IP et Telnet
  - permet des transferts dans les deux sens
  - fournir un dialogue interactif
  - n'est pas limité à Unix
  - mais n'est pas sécurisé
- les administrateurs le limitent souvent :
  - obligation de FTP passif (commandes limitées)
  - obligation de recours à `sftp` (utilisation du protocole de Ssh)
  - interdiction totale

La commande ftp

- comportement semblable à la commande `telnet` :
  - appel avec un nom de machine :
    - `ftp` *machine*
    - la tentative de connexion débute immédiatement
  - appel sans paramètre
    - affichage de l'invite `ftp>` et attente de commande
- dialogue interactif de connexion :
  - `open` *machine*
    - la machine distante affiche un message de bienvenue et demande l'identification
  - `user` *utilisateur*
    - à utiliser en cas d'identification erronée
  - `close` pour terminer la connexion sans quitter Ftp
  - `quit` pour terminer la connexion et la commande

Ftp anonyme

- des serveurs FTP publics sont des dépositaires de logiciels libres
- connexion acceptée de l'utilisateur `anonymous`
- adresse électronique comme mot de passe
- environnement d'accueil limité au service des logiciels
- aucun accès au reste de la machine
- en général accès en extraction seulement
- parfois possibilités très limitées de dépôt de fichiers
- serveurs publics intéressants :
  - `ftp.lip6.fr`
  - `ftp.inria.fr`
  - `ftp.gnu.org`
  - `ftp.gnome.org`
  - etc.

Commandes après connexion

- `pwd` comme dans un shell
- `dir` affiche le contenu du répertoire
- `cd` *`chemin`* change de répertoire distant
- `get` *`fichier`* extrait le fichier distant
- `mget` *`fichiers`* extrait à la demande plusieurs fichiers (utilisation de jokers)
- `put` *`fichier`* dépose un fichier local
- `mput` *`fichiers`* dépose à la demande plusieurs fichiers
- `lcd` *`chemin`* change de répertoire local
- `!` *`commande`* exécute localement la commande

Utilisation dans Emacs

- notation spéciale pour un nom de fichier : `/nom@machine:chemin`
- la présence des trois caractères `/` `@` `:` signale l'utilisation du mécanisme « tramp »
- le chemin distant est atteint grâce à un appel à sftp
- les mécanismes d'Emacs sont tous disponibles
- spécialement intéressant en mode Dired

Le client NcFTP

- le protocole FTP de base n'est pas aimé des administrateurs à cause de ses propriétés :
  - la connexion initiale du client vers le serveur sert seulement à transmettre des commandes
  - chaque commande d'interrogation ou d'extraction nécessite une deuxième connexion, du serveur vers le client
  - beaucoup de mécanismes de sécurité refusent cette deuxième connexion
- le client NcFTP simplifie l'utilisation du FTP passif, où cette deuxième connexion se fait du client vers le serveur
- par défaut il ouvre la connexion pour un FTP anonyme
- langage de commande similaire à celui de `ftp`
- possibilité de conserver des *marque-pages* vers les sites explorés

Clients FTP graphiques

- La commande `ftp` ou le client `NcFTP` opèrent à la ligne de commande.
- Il existe des clients FTP avec interface graphique :
  - gFTP client sous licence GNU GPL, disponible sous les systèmes UNIX. En particulier avec la `Fedora 8` et `GNOME` disponible dans les salles de TP.
  - `FileZilla` à l'origine pour `Windows` et depuis la version 3.0 multiplate-forme. Il existe également le serveur FTP du même nom.
  - Accessoirement le nagitateur `Firefox` permet d'accéder à un serveur FTP, et fait office de client à l'aide de l'`add-on` FireFTP

Accès depuis chez vous

- les mécanismes utilisés au *MIPS* (Matériel informatique pédagogique des Sciences) vous offrent un service important et varié :
  - accès à plusieurs serveurs différents
  - puissance de calcul et nombreux logiciels
  - espace sur disques
- il est donc utile d'y accéder depuis chez vous, de la manière suivante :
  - avec GNU/Linux, utilisez `ssh` ou `sftp` dans une fenêtre `xterm`
  - avec Windows, installez le logiciel libre `putty` : `http://www.chiark.greenend.org.uk/~sgtatham/putty/`
  - dans les deux cas, établissez la connexion avec `sesame-mips.unice.fr`, machine relais d'où vous pouvez vous connecter à `uranie` ou un PC par `ssh`
  - l'accès initial est lent, mais ensuite satisfaisant

Accès distant avec utilisation de X

- la connexion graphique à distance pose des problèmes de temps de réponse
- en effet, le dialogue entre les clients X et le serveur nécessite un débit important dans les deux sens
- on peut l'éviter en utilisant `vncserver`
- cela lance sur le PC distant un serveur X virtuel
- les clients sur le PC dialoguent avec ce serveur local
- depuis chez vous, l'application `vncviewer` permet de visualiser l'écran de ce serveur virtuel
- la connexion est possible par un tunnel SSH
- le site du cours donnera les détails

Extraction de fichiers non interactive

- la commande `wget` permet entre autres de :
  - extraire des fichiers par le protocole FTP
  - extraire des pages Web par le protocole HTTP
  - extraire un ensemble de fichiers
  - reprendre une extraction incomplète
  - travailler en différé et sans intervention
- forme d'appel : `wget [`*`options`*`]` *URL*
- parmi les options intéressantes :
  - passage en travail d'arrière-plan, permettant même la déconnexion
  - lecture des URLs depuis un fichier
  - reprendre une connexion interrompue
  - extraire une hiérarchie jusqu'à un certain niveau
  - etc.

Plan en cours

1 Connexion à une machine distante
- Mécanismes élémentaires
- Mécanismes sécurisés

2 Utilisation de fichiers distants
- Méthodes simples
- Méthodes perfectionnées

3 Communication hypertexte
- Le protocole HTTP
- Le langage HTML
- Les navigateurs

Le protocole HTTP

- HTTP = *HyperText Transport Protocol*
- un document *hypertexte* contient des références à d'autres documents
- un serveur HTTP :
  - fournit une base de documents hypertextes, souvent appelés des *pages*
  - reçoit des requêtes selon le protocole HTTP
- les pages sont le plus souvent des fichiers codés dans le langage HTML
- un client HTTP est capable de :
  - envoyer des requêtes concernant des pages
  - afficher le contenu des pages reçues
  - utiliser les références (ou *liens*) contenues dans les pages pour envoyer de nouvelles requêtes

Ce qu'est un lien hypertexte

- un lien hypertexte est une *URL* (*Universal Resource Locator*)
- sa forme générale est la suivante :

  protocole`://`*hôte*`[:`*port*`][/`*chemin*`/`*fichier*`]`

- exemple :

  `http://deptinfo.unice.fr/~guingne/L1MI-SI/`

  Il s'agit des documents décrivant l'enseignement de systèmes informatiques
- remarques :
  - pour le protocole et l'hôte, majuscules et minuscules sont équivalentes
  - le numéro de port n'est que très rarement indiqué
  - le chemin et le fichier respectent les conventions de Unix, donc majuscules et minuscules ne sont pas équivalentes

Le World Wide Web

- l'idée du *World Wide Web* (réseau mondial) a été conçue au CERN (Centre européen de recherches nucléaires) vers 1989
- elle est souvent confondue avec Internet
- elle repose sur le protocole HTTP
- les premiers *navigateurs* étaient purement textuels
- les navigateurs actuels acceptent également d'autres protocoles, en particulier FTP, Telnet, NNTP, Gopher et Wais
- on utilise parfois de plus les navigateurs comme outils de comunication universels pour des applications déjà vues :
  - outil de courrier
  - outil de conversation
  - outil de groupes de discussion

Fonctionnement

- il n'y a pas de session HTTP
- chaque *requête* est indépendante des autres
- le client envoie une requête GET qui précise :
  - la référence à la page demandée
  - le protocole utilisé
- le serveur connaît également (et conserve) :
  - l'adresse IP de la machine cliente
  - la date et l'heure de la requête
- la requête peut être accompagnée de plusieurs lignes supplémentaires précisant les codages acceptés, le type de navigateur, etc.
- la réponse du serveur, accompagnée éventuellement de ce qui est demandé, termine la requête

Remarque importante

- le protocole HTTP n'était pas prévu pour ce qu'on en fait actuellement :
  - il n'y a pas d'établissement de connexion durable
  - une requête HTTP reçoit une réponse, puis la prochaine requête du même client sur le même serveur nécessite à nouveau l'établissement d'une connexion

  Mais depuis HTTP 1.0 → Connection : Keep-Alive
  et avec HTTP 1.1 : connexion persistante
- on a donc ajouté un mécanisme de marqueurs (*cookies*) :
  - envoyés au client en même temps que la réponse à la requête
  - conservés dans l'espace de travail de l'utilisateur
  - contenu quelconque, déterminé par le serveur
  - récupérables par le serveur pour mémoriser l'historique des communications depuis cet utilisateur

Ce qu'est HTML

- HTML = *HyperText Markup Language*
- texte courant parsemé de *balises*, qui servent à :
  - donner des indications de structure : titres hiérarchisés, tableaux, listes, etc.
  - donner des indications de présentation : caractères gras, italiques, couleur, etc.
  - fournir des liens hypertextes
- le codage HTML n'est pas prévu pour déterminer la totalité de la présentation de la page
- celle-ci est déterminée par le navigateur
- là encore, beaucoup de sites détournent HTML de son usage normal (« site optimisé pour tel navigateur »)

Le système de balises

- les balises vont en général par paires :
  - *balise d'ouverture* `<balise>`
  - *balise de fermeture* `</balise>`
- Exemple : `<b>texte en gras</b>`
- cela permet donc des emboîtements :
  `<h1>Titre en <i>italiques</i></h1>`
- les balises ouvrantes peuvent comporter des *attributs*
- la forme générale est `clé = "valeur"`
- Exemple : `<p align="right">Exemple de paragraphe</p>`
- la présentation du texte en HTML est sans signification
- une suite d'espaces et de passages à la ligne est équivalente à un espace

Structure d'un document

- le document entre `<html>` et `</html>`
- déclaration initiale du type de document
  `<!doctype html public "-//w3c//dtd html 4.0//en">`
- *en-tête* entre `<head>` et `</head>`
- *corps* entre `<body>` et `</body>`
- l'en-tête contient principalement le *titre* de la page :
  `<title>Titre de la page</title>`
- au total, c'est un fichier de texte qu'on peut construire avec n'importe quel éditeur :
  - EMACS ou BLUEFISH font travailler sur le fichier lui-même, avec des aides d'abréviation et de coloration
  - les navigateurs offrent souvent un éditeur spécialisé qui cache le codage

Indications de structure

- les indications de structure sont les plus importantes
- en effet, elles ont un effet indépendant du navigateur et de l'environnement de l'utilisateur
- niveaux de titres :

```
<h1>Titre de niveau 1</h1>
<h2>Titre de niveau 2</h2>
<h6>Titre de niveau 6</h6>
```

- paragraphes :

```
<p>Texte du paragraphe.</p>
```

- passage à la ligne : `<br>`
- trait horizontal : `<hr>`
- note en bas de page :

```
<fn>Texte de la note.</fn>
```

Indications de structure (suite)

- trois sortes de listes et énumérations
- liste ordonnée : les entrées sont numérotées

```
<ol>
<li> entrée 1 </li>
<li> entrée 2 </li>
</ol>
```

- liste non ordonnée : marques sur les entrées

```
<ul>
<li> entrée 1 </li>
<li> entrée 2 </li>
</ul>
```

- liste de définitions : les entrées sont identifiées

```
<dl>
<dt>Terme</dt>
<dd>Définition</dd>
</dl>
```

Tableaux

- structure générale :

```
<table>
<caption>Titre du tableau</caption>
<tr>
<th>Titre 1</th>
<th>Titre 2</th>
<th>Titre 3</th>
</tr>
<tr>
<th>Titre 1a</th>
<td>Valeur 2a</td>
<td>Valeur 3a</td>
</tr>
</table>
```

Tableaux (suite)

- attributs importants :
  - `align=` (`center, left, right, justify`)
  - `valign=` (même chose)
  - `border=` largeur en pixels de la bordure
  - `cellpadding=` espace en pixels entre contenu et cadre des cellules
  - `cellspacing=` épaisseur de la grille interne
  - etc.

Indications de présentation

- les indications de présentation sont à utiliser avec modération
- en effet, on ne peut jamais complètement maîtriser la présentation
- caractères spéciaux
  - les caractères accentués même les plus ordinaires devraient être codés
  - é = `&eacute ;`
  - cela reste utile pour les caractères rares
- les balises de style donnent des indications de présentation :
  - caractères gras : `<b> ... </b>`
  - italiques : `<i> ... </i>`
  - indice : `<sub> ... </sub>`
  - etc.

Indications de présentation (suite)

- il existe aussi des indications logiques
  - forte accentuation : `<strong> ... </strong>`
  - nom de personne : `<person> ... </person>`
  - acronyme : `<acronym> ... </acronym>`
  - etc.
- de même pour des structures logiques
  - citation : `<blockquote> ... </blockquote>`
  - etc.
- on peut aussi indiquer des dimensions absolues (à déconseiller vivement) ou en pourcentage (mieux)
  - `<hr size=5 width=20% align=left>`
  - `<table width="556" border=3 cellpadding=5>`

Feuilles de style

- la méthode recommandée actuellement est de séparer les indications de présentation du contenu des pages
- la page contient des balises d'identification
- elle fait référence à une feuille de style
- cette dernière donne les indications de présentation :
  - polices et présentation des titres
  - couleurs des liens (dépendant de leur état)
  - présentation des différentes sections
  - etc.
- il est recommandé de faire valider les pages et les feuilles de style

Liens hypertextes

- un *ancrage* est une zone cliquable qui représente un lien hypertexte
- il peut amener vers
  - un autre endroit du même document
  - un fichier situé sur le même serveur
  - un fichier situé sur un autre serveur
- forme générale :

```
<a href="adresse ou URL"> ... </a>
```

- exemples :
  - URL distante : `<a href="http://unice.fr">UNSA</a>`
  - fichier local : `<a href="../index.html">Site</a>`
  - référence interne : `<a href="#signet">Ici</a>`
- le signet est défini par l'attribut `id=`

```
<p id="#signet">Texte d'un paragraphe.</p>
```

Choix du navigateur

- les *navigateurs* sont nombreux, et assez différents
- les navigateurs graphiques sont les plus connus
  - logiciels souvent énormes, souvent utilisés comme outils à tout faire
  - le plus utilisé, Internet Explorer, est un gouffre en termes de sécurité et le seul qui ne fonctionne pas sous Unix
  - la famille de Mosaic, Netscape, Mozilla et Firefox est maintenant du logiciel libre et sécurisé
  - les versions de GNU/Linux telles que Galeon ou Epiphany s'appuient sur Mozilla
  - citons encore Opera et Amaya
- le navigateur purement textuel Lynx a l'avantage d'être très simple et rapide