Gestion des processus

12.1 NOTIONS THÉORIQUES SUR LES PROCESSUS

Nous avons, tout au long de cet ouvrage parlé des processus sans jamais vraiment les définir.

Nous allons tout d'abord présenter quelques notions théoriques concernant les processus, puis les cinq modes d'exécution d'une commande sous Linux, ainsi que les commandes `ps` et `kill` qui permettent de gérer les processus. Enfin nous présentons le job control.

12.1.1 Processus

Un processus est un programme en cours d'exécution. Les attributs d'un processus appartiennent à ce que l'on appelle son environnement : parmi eux le code, les données temporaires, les données permanentes, les fichiers associés, les variables. L'environnement d'un processus contient aussi les entités que le système lui attribue : les descripteurs, la mémoire allouée, la pile d'exécution du noyau.

12.1.2 Processus père et processus fils

Le processus fils est un processus qui a été créé par un autre processus qui prend le nom de processus père.

12.1.3 Identification d'un processus

Un processus sous Linux est identifié par un numéro unique qui s'appelle le numéro d'identification du processus PID (Process IDentifier) et qui lui est attribué par le système à sa création.

12.1.4 Temps partagé

On appelle **temps partagé** une exploitation dans laquelle plusieurs processus sont simultanément en cours d'exécution. Cette simultanéité n'est qu'apparente. Le noyau Linux va affecter à l'unité centrale le processus en cours dont la priorité est la plus importante. Lorsque ce processus est en attente ou si un processus de priorité plus importante apparaît, le noyau libère l'unité centrale du processus en cours et charge le nouveau processus de haute priorité. De surcroît, afin qu'un processus de haute priorité ne monopolise pas l'unité centrale, le noyau Linux va modifier dans le temps les priorités des processus en attente et en cours afin de permettre à l'ensemble des processus d'être exécuté.

12.1.5 Swapping (va-et-vient)

Le **swapping** consiste en la recopie sur disque d'un processus complet ou d'une partie d'un processus ayant perdu le contrôle de l'unité centrale et ne pouvant plus rester en mémoire centrale. La mémoire centrale ainsi libérée est affectée à un processus plus prioritaire. Il est aisé de comprendre que la performance d'un système dépend beaucoup de la qualité du swapping.

12.1.6 Classification des processus

Il est possible de distinguer deux types de processus : les processus système et les processus utilisateurs.

a) Les processus système (daemons)

Ces processus ne sont sous le contrôle d'aucun terminal et ont comme propriétaire l'administrateur du système ou un uid d'administration. Ils assurent des tâches d'ordre général, parfois disponibles à tous les utilisateurs du système. Ils ne sont d'habitude stoppés qu'à l'arrêt du système d'exploitation. Les plus courants sont :

- `init` : initialise un processus par terminal connecté sur la machine, permettant la connexion des utilisateurs. Ce processus a le numéro 1. C'est le processus parent de tous les interpréteurs de commandes créés par la connexion d'un utilisateur.
- `crond` : permet l'exécution d'un programme en mode cyclique.
- `xinetd` : super démon internet chargé de créer les processus serveurs réseau sur requêtes des clients.

b) Les processus utilisateurs

Ils correspondent à chaque exécution d'un programme par l'utilisateur, le premier d'entre eux étant l'interpréteur de commandes à la connexion. Ces processus appartiennent à l'utilisateur et sont généralement attachés à un terminal.

12.2 EXÉCUTION D'UNE COMMANDE

Les chapitres précédents ont présenté plusieurs modes d'exécution d'une commande. En fait il existe cinq modes d'exécution d'une commande sous Linux :

- Le mode interactif (foreground),
- Le mode en arrière-plan (background), appelé aussi mode asynchrone,
- Le mode différé,
- Le mode batch,
- Le mode cyclique.

12.2.1 Le mode interactif

En mode interactif, mode le plus fréquemment utilisé, la commande est lancée à partir d'un interpréteur de commandes. Pendant l'exécution de la commande, l'utilisateur ne peut pas utiliser le terminal pour lancer une autre commande. Le contrôle n'est restitué à l'utilisateur qu'à la fin de l'exécution de la commande.

À tout moment, il est possible d'interrompre cette commande en utilisant la combinaison de touches *<ctrl c>*, ou de la suspendre à l'aide de *<ctrl z>* (attention, la fonction de ces touches peut être modifiée par la commande *stty*).

12.2.2 Le mode en arrière-plan

Le lancement de commandes en arrière-plan (paragraphe 7.4) permet de rendre immédiatement le contrôle à l'utilisateur. Cette fonctionnalité est intéressante pour des tâches ne nécessitant pas d'interaction entre l'utilisateur et la tâche, comme par exemple la compilation d'un programme.

La commande est lancée suivie du caractère **&**. Son exécution peut être surveillée par les commandes *ps* ou *jobs*.

Exemple

Le fichier *essai* est un exécutable. Son exécution n'interagit pas avec l'utilisateur.

```
xstra> essai &
xstra>
```

L'utilisateur a la main. Il peut continuer à travailler tout en surveillant régulièrement l'exécution de la commande *essai*.

Si l'utilisateur est en Bash et qu'il quitte le shell, la commande en arrière-plan est interrompue automatiquement. Pour éviter ce problème, il faut lancer la commande sous le contrôle de la commande *nohup* ou utiliser la commande interne *disown* que nous décrirons plus tard.

Exemple

```
xstra> nohup essai &
xstra>
```

12.2.3 Le mode différé

L'exécution différée d'une commande est réalisée à l'aide de la commande *at* qui permet de déclencher l'exécution d'une commande à une date fixée.

Exemple

```
xstra> at 20:05 20/09/00  <commande
xstra>
```

demandera le lancement du contenu de *commande* le 20/09/00 à 20 h 05.

L'autorisation d'utilisation de *at* pour un utilisateur est indiquée dans le fichier */etc/at.allow*. Si ce fichier n'existe pas, le fichier */etc/at.deny* est vérifié pour déterminer si l'accès à la commande *at* doit être interdite à l'utilisateur. Si *at.deny* est vide, l'utilisation de *at* est permise pour tous les utilisateurs. Si aucun fichier n'existe, seul l'administrateur (root) a la permission d'utilisation de *at*.

Par défaut *at* lance l'interpréteur de commandes *sh*. De plus le processus ainsi créé ne sera associé à aucun terminal ; les sorties sont soit redirigées soit envoyées à l'utilisateur par *mail*. Il est possible de gérer à l'aide de *atq* (ou *at l*) (liste des processus en cours avec indication du numéro d'identification) et de *atrm* (ou *at d numero at*) (suppression d'un processus) les processus lancés par un utilisateur.

12.2.4 Le mode batch

Le batch permet de placer une commande dans une file d'attente. Le système exécutera toujours la commande placée en tête dans la file d'attente. Ainsi toutes les commandes lancées par *batch* seront exécutées séquentiellement quel que soit l'utilisateur qui a mis la commande dans la file d'attente. La gestion des sorties standard est similaire à celle de la commande *at*. Ce mode est le plus altruiste sur un système Linux très chargé.

12.2.5 Le mode cyclique

L'exécution cyclique d'une tâche est réalisée à l'aide de la commande *crontab*. Le processus *cron* (daemon) scrute un fichier dans lequel sont définies les commandes à exécuter à date fixe. L'autorisation d'utilisation de *crontab* pour un utilisateur est identique à la méthode définie pour l'autorisation d'utilisation de la commande *at*. Les fichiers vérifiés sont */etc/cron.allow* puis */etc/cron.deny*.

L'utilisateur va créer un fichier comprenant les indications de répétition d'exécution de la tâche ainsi que la tâche elle-même. La commande `crontab file` va copier le fichier `file` dans la zone spool du crontab et ainsi le soumettre à une exécution cyclique.

On trouvera dans le répertoire `/var/spool/cron` un fichier au nom de l'utilisateur. Ce fichier contiendra les commandes à exécuter cycliquement.

La commande `crontab e` permet d'éditer directement votre crontab contenue dans `/var/spool/cron`.

Chaque ligne de `file` a six champs. Chaque champ est séparé par un espace ou une tabulation. La signification des champs est la suivante :

Champ 1 : la minute (0-59),

Champ 2 : l'heure (0-23),

Champ 3 : le jour du mois (1-31),

Champ 4 : le mois de l'année (1-12),

Champ 5 : le jour de la semaine (0-6 avec 0 = dimanche),

Champ 6 : la tâche à exécuter.

Chacun des champs peut être soit un astérisque (`*`) signifiant toutes les valeurs, soit une liste d'éléments séparés par des virgules, soit deux nombres séparés par un tiret ( ) indiquant une fourchette inclusive de valeurs.

L'exemple suivant :

```
59 0 * * 1 6    sauvegarde_journaliere
```

permet d'exécuter la commande `sauvegarde journaliere` tous les jours du lundi au samedi à 0 h 59. Attention, l'utilisation de `crontab` n'est possible que si l'administrateur vous le permet. Les sorties standard sont envoyées à l'utilisateur par la messagerie, sauf en cas de redirection.

12.3 LA COMMANDE PS

La commande `ps options` permet d'obtenir des renseignements sur l'état des processus en cours. Par défaut, seuls les processus de la connexion en cours pour l'utilisateur ayant lancé la commande `ps` sont affichés. Une option `u nom utilisateur` permet de sélectionner les processus de cet utilisateur pour l'ensemble de ses connexions. Parmi les nombreuses options de cette commande, les options suivantes sont particulièrement utiles :

-e affiche des renseignements sur tous les processus en cours,

-C affiche des renseignements sur tous les processus d'un certain nom :
`ps C bash`

-f génère, pour chaque processus, le nom de l'utilisateur (UID), le numéro du processus (PID), le numéro du processus père (PPID), l'heure de lancement

du processus (STIME), le nom du terminal (TTY) et le temps d'exécution du processus (TIME).

Exemples

```
xstra> ps
PID         TTY    TIME   CMD
3408        pts/1  0:00   ps
10804       pts/1  0:20    ksh
xstra> ps  ef
UID         PID  PPID  C STIME TTY          TIME CMD
root          1     0  0 May18 ?        00:00:03 init
root        300     1  0 May18 ?        00:00:08 syslogd  m 0
daemon      327     1  0 May18 ?        00:00:00 /usr/sbin/atd
root        343     1  0 May18 ?        00:00:00 crond
root        359     1  0 May18 ?        00:00:03 inetd
root        411     1  0 May18 ?        00:00:00 sendmail
root        444     1  0 May18 ?        00:00:00 httpd
xstra     11736 11731  0 13:45 pts/0    00:00:00 /bin/bash
xstra     11737 11732  0 13:45 pts/1    00:00:00 /bin/bash
xstra     11979  3955  0 16:29 ?        00:00:00 kwm
xstra     11995 11979  0 16:29 ?        00:00:00 kbgndwm
xstra     12021 11979  0 16:30 ?        00:00:02 kfm
xstra     12022 11979  0 16:30 ?        00:00:00 krootwm
xstra     12023 11979  0 16:30 ?        00:00:01 kpanel
xstra     12098 11737  0 16:41 pts/1    00:00:00 man at
xstra     12160 11736  0 17:17 pts/0    00:00:00 ps  ef
```

12.4 LA COMMANDE KILL

Cette commande *kill signal PID* sert essentiellement à arrêter des processus en arrière-plan (background). En effet, pour arrêter le ou les processus liés à la commande en cours d'exécution, il est beaucoup plus simple de taper <ctrl-c>.

L'option *signal* correspond au signal envoyé au processus. Un certain nombre d'événements (erreur, *<ctrl c>*,…) peut être signalé à un processus à l'aide d'un signal. Ce mécanisme permet la communication interprocessus, notamment entre le système d'exploitation et le processus. Les principaux signaux sont :

SIGHUP	(SIGnal Hang UP : fin du shell)
SIGINT	(SIGnal INTerrupt : interruption du programme)
SIGKILL	(SIGnal KILL : tuer le processus)
SIGTERM	(SIGnal TERMinate : terminaison douce)
SIGQUIT	(SIGnal QUIT : terminaison brutale)
SIGSTOP	(SIGnal STOP : stopper le processus)

D'autres signaux existent et sont décrits dans *signal* (7). Par défaut la fonction *kill* envoie au processus le signal SIGTERM lui demandant de s'arrêter. Un processus peut ignorer le signal SIGTERM. Le signal SIGQUIT est plus brutal. Le

signal SIGKILL ne peut être ignoré et permet dans tous les cas d'arrêter le processus (l'arrêt peut alors avoir des conséquences graves). S'il faut tuer un processus, il est donc prudent d'utiliser les commandes suivantes, dans cet ordre :

```
kill pid
kill  QUIT
kill  KILL pid.
```

12.5 LE JOB CONTROL

Le **job** est une ligne de commandes shell. Il est composé d'un ou de plusieurs processus dans le cas d'utilisation du tube (voir le paragraphe 7.3 et 7.4). Chaque job est numéroté de 1 à N par le shell. Ce numéro est interne au shell. Il est plus maniable que le PID. Un job peut se trouver dans trois états :

- **avant-plan** ("foreground"). Le job s'exécute et vous n'avez pas la main en shell.
- **arrière-plan** ("background"). Le job s'exécute et vous avez la main en shell.
- **suspendu** ("suspended"). Le job est en attente, il ne s'exécute pas.

Le passage d'un état du job à un autre est présenté sur la figure 12.1 dans le cas du Bash. Certaines fonctions de création et d'action des jobs ont déjà été présentées dans le chapitre 7. Chaque job peut être référencé en utilisant le préfix % suivi du numéro de job. Nous allons dans ce paragraphe récapituler les différentes fonctions du "job control" et les compléter.

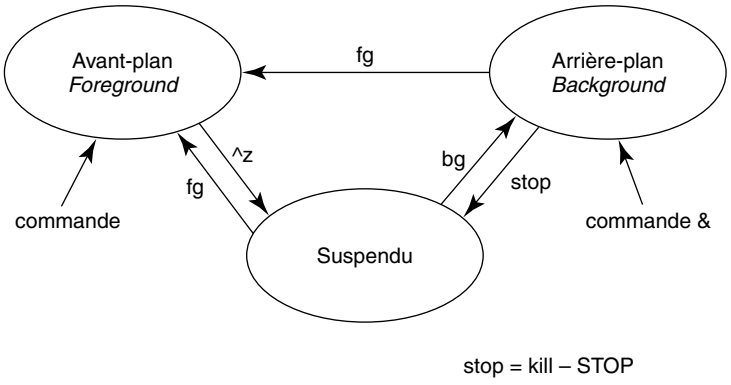

FIGURE 12.1. DIFFÉRENTS ÉTATS D'UN JOB.

12.5.1 Job en arrière-plan

La création d'un job en arrière-plan est réalisée par l'utilisation du lancement d'une commande en "background" (paragraphe 7.4).

Exemple

```
xstra> (find /usr  type d  print | sort >/tmp/toto) &
[1] 2349
xstra> (find /usr  type f  print | sort >/tmp/tutu) &
[2] 3786
xstra>
```

Les deux jobs créés dans cet exemple tournent simultanément. Chacun comporte deux processus. Le shell après chaque commande rend la main et permet l'utilisation du shell en interactif.

La commande *jobs l* permet de lister les jobs en cours.

Exemple

```
xstra> sleep 60&
[1] 2736
xstra> sleep 50&
[2] 2957
xstra> sleep 40&
[3] 3100
xstra> jobs  l
[1]    2736   Running  sleep 60&
[2]    2957   Running  sleep 50&
[3]  + 3100   Running  sleep 40&
xstra>
```

Dans l'exemple précédent, le caractère + indique le job le plus récent, le caractère – indique le deuxième plus récent.

12.5.2 Job suspendu

Une commande peut être suspendue lorsqu'elle est en avant-plan par la touche *<ctrl Z>*.

Lorsqu'elle est en arrière-plan la commande :

kill SIGSTOP %numero job ou

kill STOP %numero job

permet de suspendre la commande associée au numéro de job *numero job*.

La commande *bg %numero job* permet de basculer un job suspendu en job en arrière-plan.

Exemple

```
xstra> stty susp <ctrl Z>
xstra> sleep 100
...
```

```
<ctrl Z>
[1]  2655   Stopped
xstra> jobs  l
[1]   2655   Stopped  sleep 100
[2]   2957   Running  beta&
xstra> kill  STOP %2
[2]  Stopped beta
xstra> bg %2
xstra>
```

12.5.3 Job en avant-plan

Toute commande lancée interactivement en shell est en avant-plan. Un job en arrière-plan ou suspendu peut être mis en avant-plan par la commande *fg %numero job*.

Exemple

```
xstra> jobs  l
[1]   2655   Stopped  sleep 100
[2]   2957   Running  beta&
xstra> fg %1
  § Attente de quelques secondes,
  § la commande sleep 100 est en avant plan
xstra>
```

12.5.4 La commande kill et le job control

L'un des grands avantages du "job control" est de faciliter grandement les possibilités pour tuer un processus. En effet tuer un job est plus pratique que tuer les processus qui le constituent : il est inutile de manipuler des numéros de processus, surtout dans le cas où le job est constitué de plusieurs processus (tube, commandes groupées).

Exemple

```
xstra> (sleep 55; sleep 66)&
[1] 11877
xstra> jobs  l
[1]+ 11877 Running ( sleep 55; sleep 66 ) &
xstra> kill %1
xstra> jobs  l
[1]+ 11877 Terminated ( sleep 55; sleep 66 )
xstra>
```

12.5.5 Sortie de session et job control

Lorsqu'on quitte une session tous les processus attachés à la session seront interrompus. En effet, le Bash envoie à chaque job le signal SIGHUP indiquant la fin de

session et par conséquent l'arrêt du job. Pour éviter qu'un processus en arrière-plan soit ainsi interrompu en quittant sa session il faut :

- soit utiliser la commande *nohup* lors du lancement de la commande (voir paragraphe 7.4),
- soit désactiver l'envoi du signal SIGHUP par le Bash en utilisant la commande *disown -h %numero_job*. Cette commande n'est disponible que pour la version 2 du Bash.

Exemple

```
xstra> jobs -l
[1]   2655   Running  sleep 100
[2]   2957   Running  beta&
xstra> disown -h %2
xstra> # En quittant la session, la commande beta n'est
       # pas interrompue.
```

12.6 EXERCICES

Exercice 12.6.1

Écrivez un shell script qui cherche dans votre arborescence personnelle tous les fichiers de noms *core*, **.tmp*, *a.out* qui n'ont pas été accédés depuis plus de 3 jours, les supprime et vous envoie la liste par mail. Ce script sera exécuté tous les jours à trois heures du matin, sauf les samedi et dimanche.

Exercice 12.6.2

Écrivez le shell script `killprog` qui permet d'envoyer le signal SIGKILL à un processus désigné non pas par son PID mais par son nom : `killprog bash`. Il faudra prendre garde au fait que plusieurs processus différents peuvent porter le même nom, en présenter clairement la liste, et toujours demander confirmation avant d'envoyer le signal.

Chapitre **13**

Réseaux

13.1 INTRODUCTION

Après l'informatique centralisée, nous avons connu une période de développement intense de l'informatique personnelle : stations de travail et surtout micro-ordinateurs. Cette nouvelle informatique a bouleversé les façons de faire et les organisations informatiques traditionnelles. Mais l'évolution a été trop radicale et cette informatique personnelle s'est révélée souvent trop individualiste pour s'intégrer harmonieusement à des organisations importantes.

Si les inconvénients de la centralisation de l'informatique sont connus, les avantages de la centralisation de l'information sont évidents : comment garantir en pratique l'intégrité de l'information si elle est dupliquée sur de nombreux systèmes incompatibles ou communicant mal entre eux ?

Les réseaux locaux se sont alors développés pour résoudre ces problèmes. L'informatique est aujourd'hui très souvent distribuée, ce qui permet de bénéficier simultanément des avantages de la centralisation de l'information et de la décentralisation des moyens, et des atouts que procure une circulation rapide de l'information entre tous ses utilisateurs.

S'il n'est donc pas indispensable d'utiliser Unix/Linux pour construire un réseau (surtout homogène), il est par contre très avantageux de disposer de machines Unix/Linux pour fédérer un réseau de machines hétérogènes. Dans ce domaine aussi, Unix a été un puissant fédérateur et les protocoles développés initialement sous ce système sont supportés par toutes les plates-formes existantes. Ce fait a largement contribué au succès des systèmes ouverts.

13.1.1 Les avantages du réseau

a) Communication facile et rapide de l'information

Particulièrement importante dans le domaine de la recherche qui a vu naître les grands réseaux, la communication rapide et à grande échelle de l'information est indispensable à toute organisation dont la taille dépasse le groupe d'individus. Une organisation répartie sur plusieurs sites distants ne peut plus se passer d'un réseau d'interconnexion, quelle que soit son activité.

b) Partage de ressources : matérielles, logicielles, données

La mise en commun de ressources matérielles : imprimantes, espace disque, périphériques coûteux ou calculateurs puissants utilisés de façon épisodique, est une puissante motivation à la mise en réseau. La mise en commun de ressources logicielles procède de la même logique, et une licence logicielle, comme une imprimante, peut être partagée.

La mise en commun de données est un point essentiel au bon fonctionnement d'une organisation. Sans aller jusqu'à parler des bases de données, la centralisation et le partage de l'information permettent d'éviter les incohérences de duplication. La mise à jour non simultanée des différentes copies conduit en effet à des erreurs d'exactitude de l'information pour certains utilisateurs.

Outre ces deux points (économie de moyens techniques et amélioration de l'intégrité de l'information), cette mise en commun de ressources procure d'importantes économies de moyens humains : la mise à jour d'un jeu de données ou d'un logiciel, effectuée une fois, est immédiatement prise en compte par tous les utilisateurs de toutes les machines du réseau.

c) Accès immédiat et transparent à l'outil le plus adapté

Il s'agit simplement d'optimiser l'investissement en moyens informatiques. Par exemple, un utilisateur effectuant d'importants calculs statistiques sur de grands jeux de données peut avoir besoin d'un tableur pour visualiser ses résultats. Pourquoi utiliser la même machine pour ces deux tâches si différentes ? Si l'accès transparent à l'information est assuré, la mise en réseau de machines de puissances très différentes permet d'utiliser chacune de façon optimale, sans inconvénient pour l'utilisateur.

13.1.2 Les applications réseau

Les applications réseau représentent les fonctionnalités offertes par un réseau de machines. Elles peuvent être classées en cinq catégories présentées ci-dessous. Ces applications, dont les noms sont indiqués entre parenthèses, seront détaillées dans la suite de ce chapitre.

a) Transfert de fichiers (`ftp, tftp, rcp, scp`)

Un transfert tout électronique de fichiers entre machines distantes évite tout problème d'incompatibilité et de manipulation de média, et améliore les délais d'échanges de fichiers.

b) Connexion sur un ordinateur distant (`telnet, rlogin, rsh, ssh`)

L'utilisateur d'une machine A peut soit transformer son poste de travail en terminal d'une machine B, soit lancer des commandes sur cette machine B. Travaillant en environnement multi-fenêtre, un utilisateur pourra ouvrir des sessions simultanées sur des machines différentes sans quitter son poste de travail.

c) Courrier électronique (`mail, talk`)

De type courrier (`mail`) ou conversation interactive (`talk`), le courrier permet de gérer facilement des listes de diffusion. Si de plus il permet d'attacher à un message des documents non textuels (données binaires, exécutables,etc.), c'est un outil de communication irremplaçable.

d) Accès transparent à des fichiers distants (`NFS`)

Fonctionnalité très supérieure au simple transfert de fichiers cité plus haut : un sous-ensemble de l'arborescence de fichiers d'une machine est physiquement localisé sur le disque d'une autre machine. Du point de vue de l'utilisateur (et des logiciels utilisant ces fichiers), rien ne distingue ces fichiers distants de fichiers locaux. Cette fonctionnalité permet la mise en commun d'espace disque, mais surtout de logiciels ou de données, et s'étend hors du monde Unix, en particulier vers les systèmes Apple et Microsoft. En particulier, dans le monde Microsoft, les protocoles SMB sont très utilisés pour cette fonctionnalité de partage de fichier. Le logiciel SAMBA met en œuvre ces protocoles sur une machine Linux, et permet de partager des fichiers entre Linux et des postes de travail fonctionnant sous Microsoft Windows.

e) World Wide Web (`navigateur internet`)

Cette application est de loin la plus connue du grand public, au point de symboliser à elle seule "l'accès à Internet". Elle résulte de la fusion en une seule application de deux technologies :

- **l'hypertexte** permet d'inclure dans un document de grande taille des marques de renvoi (les liens hypertexte) vers d'autres parties du document, ou vers un autre document hypertexte.
- le **protocole http** permet d'accéder par le réseau à une page d'un document hypertexte stocké sur une machine quelconque du réseau. Ce protocole peut être considéré comme une simplification à l'extrême d'un ftp anonyme.

Dans un hypertexte au format HTML (format normalisé de description de documents sur le World Wide Web), les liens hypertextes permettent de renvoyer à un document situé sur une machine quelconque du réseau.

13.1.3 Les différentes échelles de réseaux

Il n'est pas inutile de définir quelques termes fréquemment rencontrés dans le vocabulaire des réseaux, et qui sont parfois sources d'ambiguïtés.

Le réseau local, **LAN** pour **Local Area Network**, désigne un réseau de faible extension, interconnectant sur un même médium quelques dizaines de machines. Un tel réseau est limité, dans le cas général, à un bâtiment ou un faible nombre de bâtiments.

Le réseau métropolitain, **MAN** pour **Metropolitan Area Network**, est un réseau plus étendu, constitué de plusieurs réseaux locaux interconnectés par des liaisons point à point dont les débits sont très souvent identiques à ceux des réseaux locaux.

Le réseau à grande échelle, **WAN** pour **Wide Area Network**, désigne un réseau constitué de plusieurs réseaux des types précédents, interconnectés par des réseaux de télécommunications publics. C'est le cas, par exemple, du réseau **Internet**, d'extension planétaire, et sur lequel sont connectées des millions de machines.

Les frontières entre ces différentes échelles ne sont pas toujours très claires, mais les définitions précédentes s'appliquent à la majeure partie des cas. Encore faut-il **distinguer le réseau physique du réseau logique**, ce qui n'est pas toujours facile pour l'utilisateur : l'appartenance au réseau Internet peut être pour lui une réalité plus immédiate que la limite de son réseau local. Les applications réseau, en tout état de cause, ne seront pas les mêmes suivant l'échelle : si le transfert de fichier, l'émulation de terminal et le courrier s'utilisent sur des réseaux à large échelle, le partage de fichiers ne se pratique habituellement que sur des réseaux locaux ou métropolitains.

13.1.4 Le concept client-serveur

Les applications réseau reposent sur le concept de **client-serveur**, terme qui prête parfois à confusion. Des processus "clients" utilisent sur le réseau des services assurés par des processus "serveurs". Les requêtes du client sont normalisées, de même que les réponses du serveur. En d'autres termes, client et serveur sont les deux moitiés d'une application réseau, et sont conçus pour travailler ensemble via le réseau. Le client prend l'initiative de la communication, le plus souvent lancé par un utilisateur. Le serveur, programme démarré automatiquement à la mise en route de l'ordinateur, répond aux requêtes du client. Dans la terminologie Unix/Linux, on parle de *démon (daemon)* pour désigner ces processus qui tournent constamment en attendant une requête provenant d'un client. La terminologie Microsoft Windows utilise le mot : *service*.

Il serait erroné de penser qu'un serveur est une machine : c'est un processus. Sur UNE machine Linux s'exécutent simultanément DES serveurs (et peut-être aussi DES clients). Il serait tout aussi erroné de penser que le serveur fournit des données et que le client les reçoit.

Il serait encore plus erroné d'associer serveur à "grosse machine" et client à "petite machine". Nous verrons plus loin que lorsqu'un utilisateur se connecte sur un ordinateur à partir d'un terminal X, le serveur X, qui exécute les opérations graphiques, est dans le terminal X. Les clients X, applications demandant un affichage graphique, s'exécutent sur l'ordinateur hôte. Idéalement, il faudrait éviter d'utiliser le terme serveur pour désigner une machine. Malheureusement, cet usage se répand de plus en plus, y compris dans le monde Linux.

13.1.5 Modèle Internet

L'interconnexion de machines, voire de réseaux, a été entreprise dès la fin des années 60 sous l'égide du ministère de la Défense, aux Etats-Unis. Ces travaux ont conduit à Internet, lancé en 1973 par la DARPA (Defense Advanced Research Project Agency), et finalisé en 1981.

Le **modèle Internet** est une norme d'interconnexion de réseaux. Ce réseau à grande échelle s'est mis en place à partir du début des années 1970 sous l'égide de l'Advanced Research Program Agency (précédant la DARPA). C'est dans ce cadre qu'a été introduite la notion de couches de communication.

13.2 LES PROTOCOLES INTERNET

Historiquement, les protocoles Internet ont été définis à une époque où l'université de Berkeley développait conjointement son réseau informatique et sa propre version d'Unix. La création de Berkeley Software Distribution (et son financement par la DARPA) visait à mettre à la disposition de tous les acteurs l'ensemble des développements concernant les protocoles TCP/IP. Cette intégration d'Internet à Unix BSD a largement contribué au succès de ces protocoles.

Les **protocoles Internet** (**IP, TCP** et **UDP**) sont universellement utilisés, non seulement par toutes les versions d'Unix et Linux, mais par tous les constructeurs et éditeurs de systèmes d'exploitation.

Le lecteur intéressé par ce sujet consultera avec profit l'ouvrage de Douglas Comer, TCP/IP : Architectures, protocoles, applications, (Inter-Éditions, 1992)

13.2.1 Le modèle Internet (modes connecté-non connecté)

La communication entre un client et un serveur peut se faire sous deux modes.

Dans le **mode connecté**, une connexion est établie entre la source et la destination pour que l'application puisse se dérouler. C'est le modèle de la conversation téléphonique. Le client et le serveur échangent des données et des accusés de réception concernant ces données.

Dans le **mode non connecté**, client et serveur s'échangent des messages de données sur le réseau sans vérification d'acheminement correct ni de réception par le destinataire.

Au niveau le plus bas, **Internet** fonctionne en **mode non connecté** : la source émet des paquets d'information sans établissement d'une connexion préalable, et ces paquets seront acheminés indépendamment les uns des autres jusqu'à leur destination. C'est au niveau de la couche de transport qu'un "circuit virtuel" peut être établi entre source et destination pour assurer une **communication en mode connecté**.

Dans le modèle Internet, il est possible d'assurer toute la communication en mode non connecté, ce qui présente des avantages de vitesse et de simplicité, au prix d'une

moindre sécurité dans la transmission. Le mode non connecté est encore appelé **mode datagramme**.

L'existence possible de ces deux modes de communication se traduit par deux protocoles de transport différents dans le modèle Internet : TCP et UDP.

Le modèle Internet définit les quatre couches suivantes (la couche 1 est la plus basse) :

#	Couche	Protocole	Rôle
4	Application	FTP TELNET etc.	Service universel vu par l'utilisateur, par exemple : transfert de fichiers, courrier électronique, etc.
3	Transport	TCP ou UDP	Contrôle de l'information de bout en bout, communication de processus à processus
2	Routage	IP	Acheminement des paquets de passerelle en passerelle, de machine source à machine destination.
1	Réseau physique	Ethernet,...	Transport physique de l'information

La figure suivante montre l'organisation générale du modèle Internet et l'empilement des différentes couches. Les applications peuvent être elles-même organisées en plusieurs couches, comme tout logiciel complexe. Un navigateur, par exemple, est une application cliente faisant appel à X11 et mettant en œuvre les protocoles http, ftp, et d'autres.

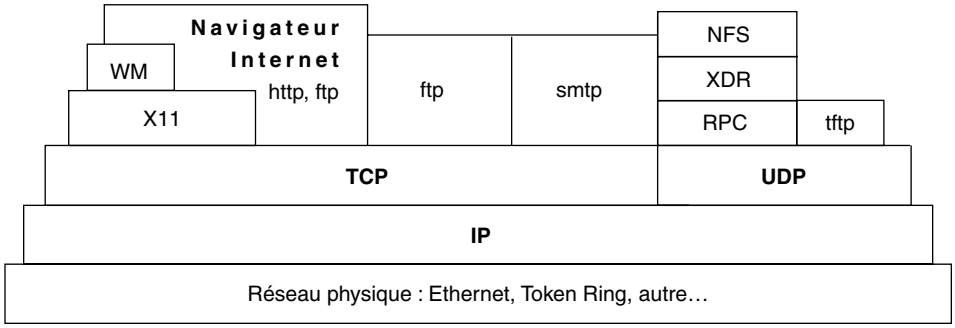

Figure 13.1. Le modèle Internet.

13.2.2 La couche réseau physique

En pratique, et dans la très grande majorité des cas, la couche réseau physique du modèle Internet est la **norme Ethernet/IEEE 802.3**. Ce n'est nullement la seule

possibilité. Du fait que l'interface entre la couche IP et la couche réseau physique est normalisée, il est possible d'utiliser comme réseau une des normes Token Ring IEEE 802.5, Token Passing Bus IEEE 802.4, X25, liaison série V24 (SLIP : Serial Line IP), ou autre.

La description d'Ethernet dépasse le cadre de ce livre, et le lecteur intéressé par le sujet peut se reporter à de nombreux ouvrages spécialisés, par exemple *Les Réseaux Ethernet d'Alexis Ferrero (Addison-Wesley)*. Pour la suite de notre exposé, il suffira de savoir qu'**Ethernet est un réseau local** ayant les caractéristiques suivantes :

- transmission par **trames** de longueur variable mais de structure normalisée
- détection des collisions avec réémission des trames perdues
- débit de 10/100 Mbit par seconde (dans une trame)
- adressage physique sur 6 octets

Sans entrer dans les détails, une **trame Ethernet** contient les champs suivants :

- préambule
- adresse Ethernet de destination
- adresse Ethernet source
- longueur de la trame
- données
- conclusion

Aucune structure n'est imposée au champ de données (sinon sa longueur maximum). Les protocoles de niveaux supérieurs, par exemple **IP**, seront donc "empaquetés" dans le champ de données des trames Ethernet. Ce mécanisme d'empaquetage est utilisé récursivement à chaque couche. Par exemple :

- les données sont transmises dans le champ de données d'un message TCP,
- ce message TCP est empaqueté dans le champ de données d'un message IP,
- ce message IP est empaqueté dans le champ de données d'une trame Ethernet.

Ce mécanisme d'empaquetage permet également de véhiculer sur un même réseau Ethernet des protocoles totalement différents, par exemple TCP/IP, Decnet et Apple, implémentant sur un même réseau physique des réseaux logiques distincts (et s'ignorant mutuellement, mais ceci est une autre histoire…).

13.2.3 La couche routage - le protocole IP Inter-network Protocol

Ce protocole résout le problème de l'acheminement de l'information dans un réseau à grande échelle. Son rôle principal est donc le routage de réseau en réseau, et il garantit l'universalité de l'adressage des machines sur le réseau Internet.

Les adresses IP

Une adresse IP (version 4) est constituée de quatre octets (seulement) ; elle est découpée en deux champs : adresse du réseau - adresse de la machine sur ce réseau.

IP définit trois classes de réseaux correspondant à des adresses ayant la forme suivante :

- réseau de classe A : 0xxxxxxx yyyyyyyy yyyyyyyy yyyyyyyy
- réseau de classe B : 10xxxxxx xxxxxxxx yyyyyyyy yyyyyyyy
- réseau de classe C : 110xxxxx xxxxxxxx xxxxxxxx yyyyyyyy

Dans chaque cas xxxx représente l'adresse du réseau et yyyy l'adresse de la machine sur ce réseau.

Les réseaux de classe A, peu nombreux, peuvent compter 16 millions de machines. Les réseaux de classe B peuvent compter 65 000 machines et les petits réseaux de classe C peuvent en compter 256.

Les adresses de réseau sont délivrées par le NIC (Network Information Center), organisme veillant à l'unicité de l'adresse et centralisant certaines informations sur les réseaux connectés à l'Internet.

Les réseaux peuvent en outre être découpés en sous-réseaux (subnet), dans ce cas l'adresse comporte trois champs :

adresse du réseau - adresse du sous-réseau - adresse de la machine

(Le champ sous-réseau ne fait pas obligatoirement 8 bits.)

Les adresses Internet sont exprimées sous forme des quatre nombres codés chacun sur un octet et séparés par un point. Par exemple :

130.79.200.1 : machine sous-réseau 200.1 réseau 130.79 (classe B)
192.93.30.5 : machine 5 réseau 192.93.30 (classe C).

Il faut noter que le terme "machine" utilisé ici désigne tout équipement directement raccordé au réseau : ordinateur, mais aussi, imprimante, matériel réseau ou terminal X (terminal graphique directement raccordé au réseau).

Il serait donc possible théoriquement de connecter 256 millions de machines sur l'Internet. En fait, le découpage en réseaux conduit à attribuer des numéros IP qui ne seront jamais utilisés, et le taux de remplissage est faible : un réseau comptant 300 machines (classe B) réserve à lui seul 65 536 adresses IP.

L'adressage sur quatre octets n'est donc pas aussi confortable qu'il pourrait sembler. Le développement exponentiel d'Internet, qui compte des millions de machines, a dépassé les espoirs de ses concepteurs.

Le routage

Une fois définie l'adresse réseau de classe A, B ou C du réseau local, une adresse IP individuelle sera attribuée à chaque machine sur ce réseau. Ces machines pourront donc échanger de l'information sur ce réseau, appelé domaine local. IP permet en outre d'atteindre des machines connectées à d'autres réseaux que le domaine local.

Pour interconnecter deux réseaux locaux (appelons-les Osiris et Cendrillon), deux solutions sont envisageables :

1) Une machine A est connectée aux deux réseaux. Cette machine a un numéro IP sur le réseau Osiris et un autre numéro IP sur le réseau Cendrillon. Cette machine est passerelle (gateway) sur chaque réseau.
2) Une machine A du réseau Osiris est reliée à une machine B du réseau Cendrillon. Cela peut être réalisé par une liaison point à point ou un réseau

public de télécommunications. A est la passerelle du réseau Osiris vers Cendrillon, B celle du réseau Cendrillon vers Osiris.

IP est responsable du routage de passerelle en passerelle jusqu'à la destination. En pratique, des **tables de routage** indiquent quelle est la passerelle à utiliser pour atteindre la destination (ou la passerelle suivante vers la destination). Il n'est heureusement pas nécessaire de disposer de tables de routage complètes sur chaque machine : une passerelle par défaut (default router) est définie dans cette table, permettant de sortir de son réseau local (son domaine) par une passerelle dont la table de routage est plus détaillée. Nous verrons dans la description des fichiers de configurations qu'il est possible de donner des noms symboliques à des adresses IP de machines et de domaines. Une adresse IP symbolique est exprimée sous la forme :

machine.domaine1.domaine2.domaine3

Les domaines IP étant hiérarchisés, le domaine Internet est organisé en sous-domaines, eux-mêmes organisés en sous-domaines, etc. Le premier niveau de sous-domaine représente le pays (fr désigne la France, uk la Grande-Bretagne) ou des grandes organisations aux États-Unis (mil pour militaire, edu pour éducation, com pour sociétés commerciales). Le niveau suivant représente souvent des réseaux métropolitains, et ainsi de suite jusqu'à la machine.

L'exemple donné précédemment en adresse IP absolue

```
130.79.200.1
 § machine sous réseau 200.1   réseau 130.79 (classe B)
```

devient en adresse IP symbolique :

```
ns1.u strasbg.fr
```

machine `ns1` dans le domaine `u strasbg`, lui-même dans le domaine `fr`.

Les passerelles peuvent être des machines Unix/Linux ; ce sont souvent des équipements spécialisés appelés routeurs. Ces derniers peuvent effectuer, à haute vitesse, le routage de protocoles différents entre des réseaux de technologies différentes, ainsi que des opérations réseau telles que le filtrage sur protocoles, sur adresses ou sur applications. Comme nous l'avons déjà signalé, la transmission au niveau IP se fait :

- de machine à machine, et non de processus à processus,
- en mode datagram (non connecté).

IP ne garantit donc pas :

- que les trames émises arriveront à destination,
- que les trames émises n'arriveront à destination qu'en un exemplaire,
- que les trames arriveront dans l'ordre dans lequel elles ont été émises.

Ces vérifications sont laissées à la charge d'un protocole de niveau supérieur. Par exemple, dans une connexion TCP, l'émetteur chargera IP de transmettre des trames

de données, le récepteur chargera IP de transmettre des trames d'accusés de réception.

13.2.4 La couche transport

Le niveau IP établit la communication entre machines. La couche transport établit la communication entre processus. Les messages de cette couche seront empaquetés dans des messages IP. L'identification des processus communicant entre eux au niveau de cette couche se fait par des numéros appelés **ports**.

Par exemple, l'application `ftp` de transfert de fichier utilise le port 21. Cette information permet le multiplexage au départ et le démultiplexage à l'arrivée des messages concernant des applications différentes, véhiculés par IP.

Le protocole UDP : User Datagram Protocol

Ce protocole permet l'échange d'information en mode non connecté. Son avantage est sa simplicité (et donc sa vitesse d'exécution). La fiabilité de transmission offerte par UDP n'est pas meilleure que celle offerte par IP. Son usage sera donc réservé à des communications locales pour lesquelles la fiabilité du réseau sous-jacent est très bonne, au transfert de messages indépendants les uns des autres (pas d'ordre de transmission), aux cas où la simplicité est le critère principal.

Par exemple, UDP est utilisé par l'application **tftp : Trivial File Transfer Protocol**, elle-même utilisée par les terminaux X (ou les stations sans disque) lors de leur mise en route (boot sur le réseau). Dans cette phase de démarrage, la simplicité est un atout important. UDP est également utilisé par NFS (Network File System de Sun) : le montage de fichiers partagés sur un réseau ne se fait pas à grande distance, et la vitesse est un critère essentiel.

Le protocole TCP : Transmission Control Protocol

Comme son nom l'indique, ce protocole est destiné au contrôle de la transmission. Établissant un circuit virtuel entre émetteur et récepteur, il instaure un dialogue en mode connecté permettant, par des mécanismes d'accusé de réception et de réémission après temporisation, la vérification de la transmission effective, unique et ordonnée des messages échangés par les applications. La communication est donc fiable, mais au prix d'une plus grande complexité et d'une charge plus importante, aussi bien sur les machines que sur le réseau.

13.2.5 Les applications réseau

Les applications réseau disponibles sur TCP/IP peuvent être classées en trois groupes :

- applications de base,
- applications étendues,
- applications Unix.

a) Les applications de base

Il s'agit d'applications existant sur tout système supportant Internet, quels que soient le système d'exploitation et le réseau utilisés. Ce sont les applications de base des réseaux hétérogènes. Il faut toutefois noter que si le système d'exploitation n'est pas multi-tâche, seuls les clients seront installés, les serveurs correspondant ne pouvant l'être. Il s'agit de :

- tftp (Trivial File Transfer Protocol)

Transfert de fichiers en mode datagram (UDP) : utilisée par les terminaux X et les stations de travail sans disque lors de la phase de démarrage (boot sur le réseau), l'application *tftp* pourrait être utilisée pour des transferts de fichiers dans le cas général.

Cette commande peut poser des problèmes de sécurité : aucun mécanisme d'identification n'est mis en œuvre. Il est possible de sécuriser le serveur tftp d'une machine Linux, en restreignant ses possibilités d'accès à une partie de l'arborescence de fichiers. Dans la plupart des cas, si la fonctionnalité serveur de boot pour des terminaux X ou des stations sans disque n'est pas nécessaire sur une machine, il est préférable de ne pas installer le serveur tftp sur cette machine (pour raisons de sécurité).

- ftp (File Transfer Protocol)

Transfert de fichiers en mode connecté (TCP) : cette commande négocie une connexion avec identification de l'utilisateur sur la machine distante : si un utilisateur A sur la machine alpha échange par *ftp* des fichiers avec un utilisateur B sur la machine beta, cet utilisateur devra connaître le login et le mot de passe de B sur beta. Pour des raisons de sécurité évidentes, ce ne peut être que la même personne (pouvant avoir des noms de login différents sur alpha et beta).

Il existe une exception très intéressante à cette règle : le **ftp anonyme** (**anonymous ftp**), permettant à l'utilisateur A sur alpha de lire (mais pas d'écrire, sauf cas particuliers) des fichiers se trouvant sur la machine beta sans identification préalable. Cela suppose, de la part de l'administrateur de beta, un travail de configuration supplémentaire du serveur ftp pour que cette connexion anonyme ne pose pas de problèmes de sécurité.

Cette possibilité est très répandue sur le réseau Internet : certains centres informatiques mettent ainsi à la disposition de la communauté Internet des informations sur l'état de leurs programmes de recherche, etc. C'est également par ce moyen qu'est diffusé le logiciel libre, dont Linux.

Le client *ftp* est lancé par la commande :

```
ftp [option] [nom_de_machine]
```

Les options ne seront pas explicitées ici (utilisez *man ftp* ou *info ftp* pour en savoir plus). Le champ *nom de machine* est une adresse Internet absolue ou symbolique, par exemple *130.79.68.78* ou *happy.u strasbg.fr*. Si ce

champ est omis, le client *ftp* passe immédiatement en mode commande. Dans le cas où le champ *nom de machine* est spécifié, la connexion est établie, l'utilisateur s'identifie sur la machine distante, puis *ftp* passe en mode commande (le prompt est *ftp>*).

Exemple

Ceci illustre un transfert par *ftp* : l'utilisateur xstra sur la machine courlis va lire un fichier sur la machine beta.

```
xstra> ftp beta.u strasbg.fr
Connected to beta.u strasbg.fr.
220 beta FTP server (SunOS 4.1) ready
Name (beta.u strasbg.fr:xstra):
331 Password required for xstra
Password:
230 User xstra logged in
ftp> ls
200 PORT command successful
150 ASCII data connection for /bin/ls
Xncd14c
Xncd15b
26 ASCII Transfer complete
ftp> binary                    § préparation pour un
200 Type set to I              § transfer de type binaire
ftp> get Xncd14c tempo         § transfert du fichier
                               § Xncd14c de beta
200 PORT command successful.   § dans le fichier
                               § tempo de courlis
150 Binary data connection for Xncd14c (130.79.130.116,1479)
226 Binary Transfer complete.
1317068 bytes received in 2.543 seconds (505.8 Kbytes/s)
ftp> quit
221 Goodbye
/xstra> ls                 § vérification sur courlis
adm/ bin/ tempo tools/     § le fichier tempo existe
xstra>
```

Attention

ftp établit une connexion avec identification sur le site distant, mais il ne s'agit pas d'un login (une session de travail). Pour ce faire, il faut utiliser *telnet*, *rsh* ou *ssh*.

Si votre machine Linux est connectée à l'Internet, il est très imprudent d'utiliser *ftp*, sauf pour des sessions ftp anonymes : le mot de passe est transféré en clair sur le réseau. Dans ce cas, *sftp* doit être préféré à *ftp*.

➤ telnet (TErminaL NETwork protocol)

Émulation de terminal sur le réseau : cette application permet la connexion à distance. Tout ce qui a été dit sur la connexion et l'identification sur le site distant à propos de *ftp* est applicable à *telnet* (sauf, bien sûr, la connexion anonyme !). Le client *telnet* est lancé par la commande :

```
telnet [nom_de_machine]
```

Le champ *nom de machine* est une adresse Internet absolue ou symbolique, par exemple *130.79.68.78* ou *happy.u strasbg.fr*. Si ce champ est omis, le client *telnet* passe immédiatement en mode commande (le prompt est telnet>). Dans le cas où le champ *nom de machine* est spécifié, la connexion est établie, l'utilisateur s'identifie sur la machine distante et entre en session sur cette machine.

Exemple

Ceci illustre une connexion à distance par *telnet*.

```
xstra> telnet beta.u strasbg.fr
Trying...
Connected to beta.u strasbg.fr
Escape character is '^T'
SunOS UNIX (beta)
login:xstra
Password:
Last login: Wed Mar 3 16 :48 :57 from alpha.u strasbg.fr
SunOS Release 4.1.2 (BETA) #1 : Mon Nov 30 10:21:04 GMT 1992
<beta>/home/xstra>ls
Xncd14c Xncd15b
<beta>/home/xstra>logout
Goodbye...
Connection closed.
xstra>
```

Il est possible de passer en mode commande par un mécanisme d'échappement (<ctrl-t> dans l'exemple ci-dessus), le prompt de *telnet* étant telnet>.

Dans le cadre de cet ouvrage, nous n'irons pas plus loin dans la description de telnet. La commande Linux *man telnet* (ou la commande help de *telnet*) permet d'en savoir plus sur les possibilités de votre implémentation de cette application très simple d'utilisation.

Si votre machine Linux est connectée à l'Internet, il est très imprudent d'utiliser *telnet* : le mot de passe est transféré en clair sur le réseau. Dans ce cas, *ssh* doit être préféré à *telnet*. Dans toutes les distributions récentes de Linux, le serveur *telnet* n'est pas activé, mais tout utilisateur peut lancer un client *telnet*.

➤ smtp (Simple Mail Transfer Protocol)

Utilisée pour l'échange de messages ascii par courrier électronique, cette application n'est pas utilisée directement, mais par l'intermédiaire de la commande *mail*

ou d'un autre client de messagerie tel que *elm*, *pine*, *mutt*, *kmail*, *exmh*, *netscape*,...

b) Les applications étendues

Ces applications, de haut niveau, ne font pas partie de TCP/IP, mais ont été développées sur ces protocoles. Elles ne sont pas limitées au monde Unix/Linux, et sont disponibles sur d'autres systèmes d'exploitation. Elles ont largement contribué au développement des réseaux hétérogènes. Il s'agit de :

- RPC Remote Procedure Call (UDP)

Ce protocole permet l'exécution de procédures sur une machine distante. Equivalent à distance d'un appel conventionnel de procédure, ce mécanisme est dit synchrone : le processus en cours attend la fin de l'exécution de la procédure distante pour continuer. Ce protocole est utilisé par des applications de plus haut niveau, en particulier par NFS.

- NFS Network File System (UDP)

NFS, développée par Sun, est universellement utilisée au point d'être le standard de fait. Ce type d'application permet de greffer une partie de l'arborescence de fichiers d'une machine alpha en un point de l'arborescence d'une machine beta. La machine alpha « exporte » un répertoire, la machine beta le monte sur un répertoire de sa propre arborescence. (Cette opération de montage peut être effectuée par plus d'une machine.)

Exemple

La machine alpha exporte le répertoire */usr/share/lib*. La machine beta monte ce répertoire sur son répertoire local */usr/lib*. Le fichier */usr/share/lib/naglib.a* de alpha est accessible sous le nom */usr/lib/naglib.a* sur la machine beta, et rien ne le différencie d'un fichier local.

Les opérations d'export et de montage sont effectuées par l'administrateur, qui peut restreindre les droits d'accès aux répertoires exportés. Ce point ne sera pas détaillé ici.

- http Hyper-Text Transfer Protocol (TCP)

Protocole utilisé pour l'accès à des pages hypertextes au format HTML. Le client http (par exemple Netscape) d'une machine demande au serveur http (par exemple Apache) d'une autre machine de lui transmettre une page. En première analyse, une URL comme :

`http://cigale.u strasbg.fr/Doc/README.html`

décrit le protocole (ici, *http*),
l'adresse IP du serveur (ici, *cigale.u strasbg.fr*)
et le nom de fichier de la page recherchée (ici, */Doc/README.html*)

La connexion TCP est établie à la demande du client, utilisée pour le transfert de cette page, puis refermée. Du point de vue du principe de base, ce protocole peut être considéré comme une simplification à l'extrême d'un *ftp* anonyme. L'interprétation du contenu de la page est à la charge du client, qui va y repérer de nouvelles URL, les présenter à l'utilisateur de la façon décrite par le format HTML de la page, et la navigation continue...

➤ X11 - X Window (TCP)

Cette application sera présentée au chapitre 16. C'est une application réseau développée sur TCP. Lorsque X Window est utilisée entièrement sur une machine, les clients X locaux accèdent au serveur X local de la même façon qu'ils le feraient pour un serveur X distant.

c) Les applications Unix

Variantes à distance de commandes Unix standard, les remote commands sont des commandes habituelles précédées de la lettre **r** (pour **remote**). Toutes ces commandes nécessitent évidemment l'identification de l'utilisateur sur la machine distante. Pour toutes les remote commands, cette identification repose sur des fichiers de configuration réseau que nous décrirons plus loin.

➤ rlogin (Remote LOGIN)

```
rlogin nom_de_machine [  l nom_login]
```

Cette application est fonctionnellement équivalente à telnet, mais de machine Unix à machine Unix (Linux). L'option *l nom login* permet de spécifier sous quel nom l'utilisateur se connecte à la machine distante ; sinon le nom local est utilisé.

rlogin présente par rapport à *telnet* deux avantages : une partie de l'environnement est transmis à la machine distante, et l'identification sur la machine distante peut être automatisée sans que la sécurité en soit gravement affectée (au contraire). En pratique, pour des raisons trop longues à détailler, l'éditeur *vi* fonctionnera souvent mieux si la connexion est établie par *rlogin* plutôt que par *telnet*.

Exemple

L'identification n'est pas automatisée (*rlogin* est la seule « remote command » pouvant fonctionner sans identification automatique).

```
xstra> rlogin beta  l platane
platane's Password :  § entrée du mot de passe sans écho
Welcome to IBM AIX Version 3.2
Last login : Mon Feb 22 10 :33 :22 1993 on pts/3 from delta.osi
ris.fr
<beta>/home/platane> ls  F
bin/ lib/ mailbox/ tools/
<beta>/home/platane> exit
Connection closed
xstra>
```

➤ rcp (Remote CoPy)

rcp permet la copie de fichiers entre la machine locale et une machine distante, ou entre deux machines distantes.

```
rcp source destination
rcp  r répertoire_source répertoire_destination
```

Cette extension de la commande *cp* nécessite, pour la désignation des noms de fichiers ou de répertoires à copier, la définition de la machine sur laquelle ils se trouvent :

```
utilisateur@nom_de_machine:nom_de_fichier
```

Le champ *utilisateur@* peut être omis si le nom de l'utilisateur est le même sur les deux machines.

Exemple

```
xstra> hostname § l'utilisateur xstra est
courlis         § connecté à la machine courlis
xstra> rcp .cshrc beta:fich1
xstra> rcp  r beta:~/tools tmp
xstra> cd tmp   § vérification
xstra> ls -F    § le répertoire tools
tools/          § a bien été copié
xstra>
```

Pour pouvoir utiliser les caractères de génération des noms de fichiers sur la machine distante, il est nécessaire de les placer entre '…' (paire de "simple quotes"), pour éviter leur interprétation localement.

Exemple

```
xstra> rcp 'beta:tools/w*' tmp
xstra> cd tmp
xstra> ls  F
where* where.bak* where.bsh* where.ksh*
xstra>
```

➤ rsh (Remote shell)

rsh permet de lancer une commande Linux sur une machine distante.

```
rsh nom_de_machine [  l nom_login] [commande]
```

L'option *l nom login* a le même sens que pour *rlogin*. Si la commande est omise, *rsh* est équivalent à *rlogin*.

Exemples

```
xstra> rsh beta ls
Mail
```

```
backup
bin
fich1
sendmail
tools
xstra> rsh beta pwd
/home/xstra
/xstra>
```

Si votre machine Linux est connectée à l'Internet, il n'est pas très prudent d'utiliser les applications *rlogin, rsh, rcp* : dans ce cas, *ssh* et *scp* fournissent exactement les mêmes fonctionnalités, avec la même syntaxe, mais en établissant entre les deux machines un tunnel chiffré évitant tout risque d'intrusion ou d'attaque. Dans toutes les distributions récentes de Linux, par souci de sécurité, les serveurs *rlogind* et *rshd* ne sont pas démarrés dans la configuration par défaut (pas plus que les serveurs *telnetd* et *ftpd*).

13.2.6 Les fichiers associés au réseau

a) Les fichiers de configuration globale

Ces fichiers concernent l'administrateur système et nous n'entrerons pas dans le détail de leur description. Signalons simplement l'existence des fichiers suivants :

/etc/hosts

Ce fichier contient une liste de couples (numéro IP, nom symbolique). C'est par son intermédiaire que IP transforme un nom symbolique de machine en une adresse IP.

/etc/resolv.conf

Dans un grand réseau, il serait fastidieux (voire impossible) de garantir que le fichier */etc/hosts* est à jour sur chaque machine. Il est alors possible de définir un "serveur de noms" (DNS : Domain Name Server), sur lequel une base de données fonctionnellement équivalente à un fichier */etc/hosts* est soigneusement maintenue à jour. Le fichier */etc/resolv.conf* décrit la liste des serveurs de noms disponibles et leurs adresses IP.

/etc/hosts.equiv

Ce fichier permet de déclarer que plusieurs machines du réseau sont équivalentes. Des utilisateurs ayant des noms identiques sur des machines déclarées équivalentes n'ont pas besoin de s'identifier lors d'une session `rlogin`, `rcp` ou `rsh`. C'est une facilité, mais également un trou potentiel de sécurité (à n'utiliser qu'en toute connaissance de cause).

/etc/services

Ce fichier contient la liste des services réseaux éventuellement disponibles sur cette machine : nom - numéro de port - protocole.

Exemple

```
# Network services, Internet style
ftp      21/tcp
telnet   23/tcp
smtp     25/tcp   mail
tftp     69/udp
```

b) Les fichiers de configuration personnelle

Ces fichiers sont créés par l'utilisateur pour configurer son propre environnement réseau.

$HOME/.rhosts

Sur la machine courlis, le fichier */home/xstra/.rhosts* contient :

```
beta.u strasbg.fr xstraplus
```

L'utilisateur xstra de courlis déclare faire confiance à l'utilisateur xstraplus de beta. Dans ce cas, une remote command lancée sur la machine courlis par l'utilisateur xstraplus de la machine beta sera exécutée sans identification préalable. Il est bien évident que l'utilisateur xstraplus de beta est la même personne que l'utilisateur xstra de courlis. Il faut noter que cette confiance n'est pas automatiquement réciproque : l'utilisateur xstraplus de beta peut ne pas avoir créé de fichier *$HOME/.rhosts*. Les permissions de ce fichier doivent impérativement être : *rw*

Ce fichier établissant la confiance n'est normalement pas suffisant pour une utilisation de *ssh*. Pour que la confiance soit établie, le fichier *$HOME/.ssh/authorized keys* doit contenir la clef de l'utilisateur client. La description des mécanismes de sécurité mis en œuvre dans *ssh* dépasse largement le cadre de cet ouvrage. Pour plus d'information, le lecteur intéressé devra se reporter à la commande : *man ssh* ou consulter le site : *http://www.openssh.com/*

$HOME/.netrc

Ce fichier est l'équivalent du précédent pour *ftp* et *telnet*; il contient des mots de passe en clair ! NE L'UTILISEZ JAMAIS, sauf pour automatiser des transferts ftp anonymes, et dans ce cas seulement.

$HOME/.forward

Ce fichier permet de rediriger les messages vers un autre destinataire ou, lorsque l'utilisateur possède un compte sur des machines différentes, de rediriger ses messages sur le compte de sa machine "principale". Il est situé dans le répertoire d'accueil ($HOME) des utilisateurs qui souhaitent renvoyer leurs messages dans une autre boîte aux lettres. Les permissions de ce fichier doivent impérativement être : *rw*

Si les protections de ce fichier ou de votre répertoire d'accueil ne sont pas prudentes, un autre utilisateur peut détourner votre courrier.

Exemple

```
xstra> whoami
jpaul
xstra> more .forward
\jpaul
xstra@courlis
```

L'utilisateur *jpaul* redirige la totalité des messages reçus vers la boîte aux lettres de l'utilisateur *xstra* de la machine courlis, tout en en conservant une copie localement (ligne \ jpaul).

13.3 EXERCICES

Exercice 13.3.1

Dans une équipe disposant de plusieurs machines Unix/Linux reliées en réseau, un utilisateur a pour nom de login pierre sur la machine courlis et pierrot sur la machine sapin. La machine sapin est équipée d'une unité de sauvegarde sur cartouche, mais pas la machine courlis. Comment fait cet utilisateur pour sauvegarder sur l'unité de sauvegarde de sapin toute son arborescence personnelle de la machine courlis.

Exercice 13.3.2

Récupérez sur le ftp anonyme de leonardo.u-strasbg.fr dans le *répertoire /pub/livre linux/exemples* le fichier tout.tar.gz.

Exercice 13.3.3

Man multi-plateforme.

Une équipe dispose de plusieurs machines unix de constructeurs différents, dont Linux, et l'utilisateur Pierre Colin dispose d'un login sur chacune de ces machines. Son nom de login n'est pas le même sur chaque machine. Cet utilisateur est enregistré sur les machines suivantes :

```
machine :  login   : Version d'Unix :
----------+----------+-----------------
courlis   : pierre  : HP-UX
sapin     : pierrot : SUN Solaris
mickey    : pierre  : IBM AIX
dingo     : pcolin  : Linux
```

Les commandes de même nom n'ont pas exactement les mêmes options et comportements sur ces différents Unix, et le *man* de chaque machine précise exactement leur fonctionnement. L'utilisateur pierre travaille normalement sur mickey, mais il veut pouvoir simplement consulter le *man* sur les autres machines, sans avoir à suivre une fastidieuse procédure de login. Il aimerait disposer des commandes suman, hpman, liman, lui permettant de consulter le man respectivement de SUN, HP et Linux lorsqu'il travaille sur sa machine habituelle.

man ls : le man de ls sur la machine locale (AIX)

suman ls : le man de ls sur la machine SUN (sapin)

hpman ls : le man de ls sur la machine HP (courlis)

liman ls : le man de ls sur la machine Linux (dingo)

Écrivez ces commandes.

Rassurez-vous : la solution est beaucoup plus courte que l'énoncé.

Exercice 13.3.4

Ftp en différé.

Si votre machine Linux est directement connectée à l'Internet, vous pouvez souvent constater que les temps de transfert par *ftp* peuvent devenir très longs à certaines heures de la journée. Il est très intéressant de lancer ce *ftp* à des heures moins chargées, ce qui est possible par la commande at. Comment faites-vous pour récupérer dans 8 heures et par un *ftp* anonyme, le fichier texte */pub/README* et le fichier binaire */pub/shell/prog.tar.gz* sur la machine ftp.machine.domaine et placer ces deux fichiers dans le répertoire */home/florent/bidon* ?