version 2009-09-18:285

Délégation Rhône-Auvergne
Formation Permanente

Administration de postes Linux en réseau

Document :
- pour les années antérieures à 2008 : Thierry Ollivier (IPNL/IN2P3) et Pierre Larrieu (CC IN2P3)
- à partir de 2008 : Denis Pugnère (IPNL/IN2P3)

Institut de Physique Nucléaire de Lyon, du 13 au 17 septembre 2010

Table des matières

Index des illustrations

1 Introduction

Ce stage consiste à appréhender l'ensemble des fonctionnalités à mettre en œuvre afin d'administrer un parc de machines *Linux/UNIX* connectées en réseau.

Nous traiterons les différentes notions nécessaires à l'administration *Linux/UNIX*. On commencera par l'acquisition de connaissances sur les réseaux IP, puis seront abordées les méthodes d'installation de distributions Linux. Nous aborderons ensuite la configuration des systèmes Linux : les commandes utilisées pour diagnostiquer ou configurer un système, les fichiers à modifier pour configurer un service.

L'intérêt d'interconnecter des machines *Linux/UNIX* est qu'elles puissent partager des ressources et coopérer : partage de disques, de périphériques telles les imprimantes, de bases d'informations tels les comptes utilisateurs, l'échange de messages ou courriers, l'acceptation de connexions distantes, etc...

L'ensemble de ces fonctionnalités s'appuie sur l'utilisation d'un protocole réseau entre les différents systèmes *Linux/UNIX*. Celui-ci est le même sur l'ensemble des différents *Linux/UNIX* et s'appelle *TCP/IP*. Il est de base, et historiquement, le protocole réseau sur *UNIX* mais son usage dépasse largement le cadre *d'UNIX* et il est présent sur de nombreux autres systèmes d'exploitation.

Le but de ce stage n'est pas de faire un cours réseau mais bien d'administrer les outils de *Linux/UNIX* qui vont utiliser *TCP/IP*. Le cours est clairement ciblé vers des administrateurs systèmes et non des utilisateurs de commandes faisant appel au réseau tels *ssh* ou *ftp*. Nous aborderons néanmoins *ssh*.

Bien qu'il ne s'agisse pas d'un cours réseau, il nous a paru opportun de rappeler dans un premier chapitre quelques notions de base sur IP.

2 Les réseaux

2.1 Un peu de théorie

Objectifs des réseaux
- Partage de ressources (programmes, données, équipements)
- fiabilisation (ex : duplication de fichiers, de serveurs)
- Réduction des coûts (un serveur, plusieurs clients) : modèle client-serveur
- Nouveau média de communication (envoi d 'informations sur de grandes distances)

Un réseau est constitué d'au moins 2 appareils capables de communiquer et d'un lien entre les appareils. Le terme « appareils » est utilisé au sens large car on trouve des machines capables de communiquer avec d'autres seulement sur 10cm, 1m, 100m 1km ... D'autre part, on trouve des liens physiques (constituant les réseaux) capables de transmettre des informations sur des distances variables allant de 3 mètres maximum (bus *USB*), 100 mètres pour l'*ethernet 100 Base T*, plusieurs kilomètres pour la fibre optique...

Ces différents types de réseaux mettent en œuvre des protocoles (ensemble de règles et descriptions qui décrire comment réaliser une action) et des technologies différents et adaptés à chaque cas.

Il existe plusieurs topologies de réseaux :

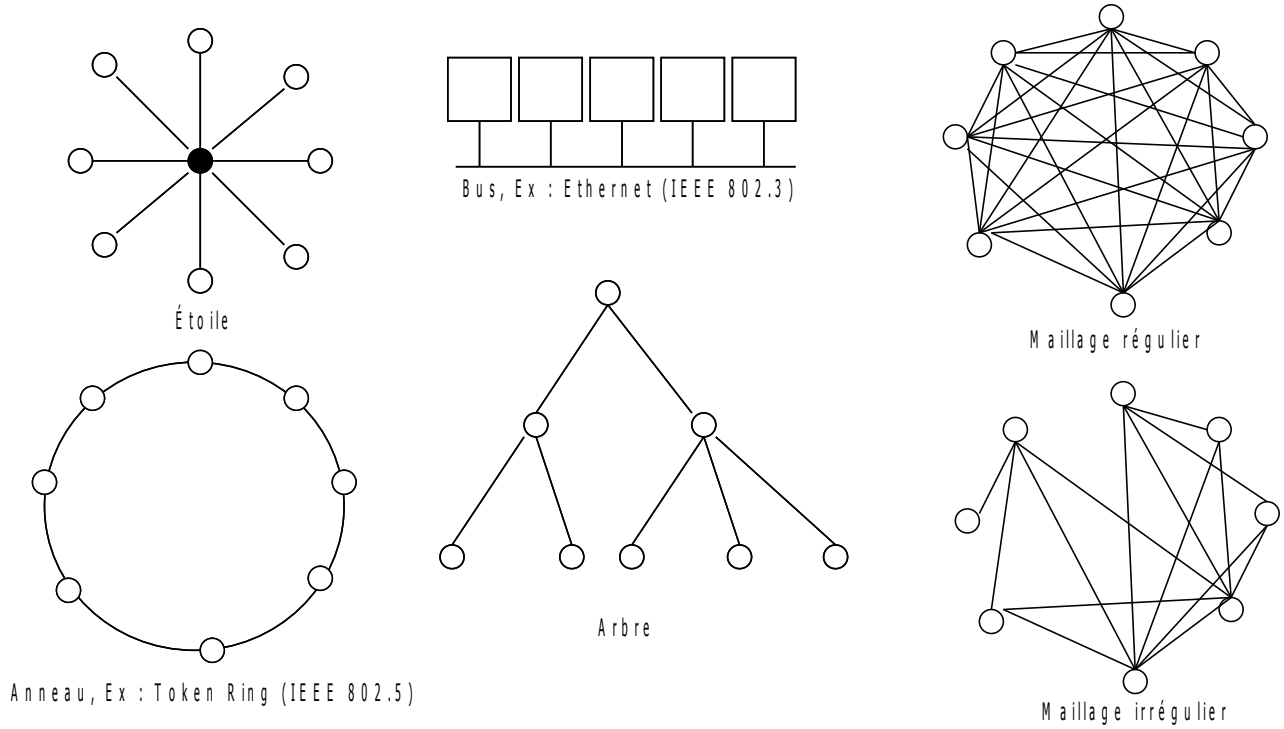


Illustration 1: Les topologies de réseaux

Plusieurs types de diffusion :
- Point à point : *Unicast* : l'information est envoyée à un seul destinataire
- Générale (*broadcast*) : une information est envoyée à toutes les autres machines du réseau local

- Restreinte *(multicast)* : une information est envoyée à un sous-ensemble des machines du réseau

Plusieurs types de communication :
- Sans connexion :
  - chaque message est indépendant,
  - chaque message est acheminé vers le destinataire indépendamment des autres,
  - arrivée dans le désordre possible,
  - chemins distincts pris par chaque message,
  - la fiabilisation de la communication par acquittements est possible, mais elle est à la charge des applications,
- Avec connexion
  - la communication se passe en trois temps :
    - établissement connexion
    - dialogue / monologue
    - fermeture connexion
  - séquencement des messages (même si chaque paquet individuel peut prendre un chemin différent des autres, le séquencement fait que le message est reconstitué dans l 'ordre)
  - la fiabilisation par acquittements est incluse par le protocole.

On va comparer les différents moyens de communication de la vie courante : Trouver si ces communication sont avec ou sans connexion ?
- Téléphone ?
- lettre postale ?
- Fax ?
- transfert de fichiers ?
- Lettre avec accusé de réception ?
- Flux multimédia (sons + images) ?

2.2 Normalisation

International Organization for Standardization, 1947, basé en suisse, (http://www.iso.ch) est un organisme dépendant de *l'ONU* et est composé de 140 organismes nationaux de normalisation. Les représentants nationaux sont des organismes nationaux de normalisation : *ANSI* pour les *USA*, *AFNOR* pour la France, *DIN* pour l'Allemagne, *BSI* pour le Royaume Uni, *HSC* pour le Japon. Ces organismes développent les standards pour faciliter l'échange de biens et services et développer la coopération.

Un des standards les plus publié est le modèle en couches de *l'ISO*. Ce modèle est appelé *OSI* (*Open Systems Interconnection*), il décrit les concepts utilisés et la démarche suivie pour normaliser l'interconnexion de systèmes ouverts (un réseau est composé de systèmes ouverts lorsque la modification, l'adjonction ou la suppression d'un de ces systèmes ne modifie pas le comportement global du réseau).

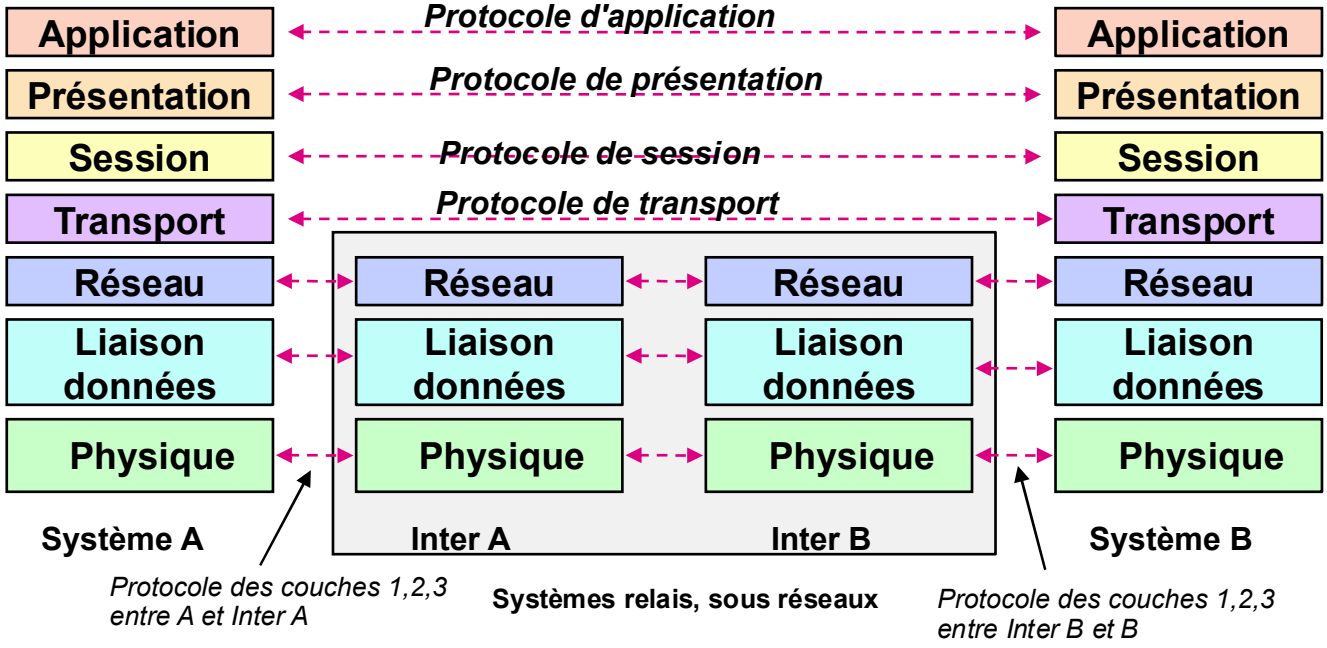
Au moment de la conception de ce modèle, la prise en compte de l'hétérogénéité des équipements était fondamentale. En effet, ce modèle devait permettre l'interconnexion avec des systèmes hétérogènes pour des raisons historiques et économiques. Il ne devait en outre pas favoriser un fournisseur particulier. Enfin, il devait permettre de s'adapter à l'évolution des flux d'informations à traiter sans remettre en cause les investissements antérieurs. Cette prise en compte de l'hétérogénéité nécessite donc l'adoption de règles communes de communication et de coopération entre les équipements, c'est à dire que ce modèle devait logiquement mener à une normalisation

internationale des protocoles.

Le modèle *OSI* n'est pas une véritable architecture de réseau, car il ne précise pas réellement les services et les protocoles à utiliser pour chaque couche. Il décrit plutôt ce que doivent faire les couches. Néanmoins, *l'ISO* a écrit ses propres normes pour chaque couche, et ceci de manière indépendante au modèle, i.e. comme le fait tout constructeur.

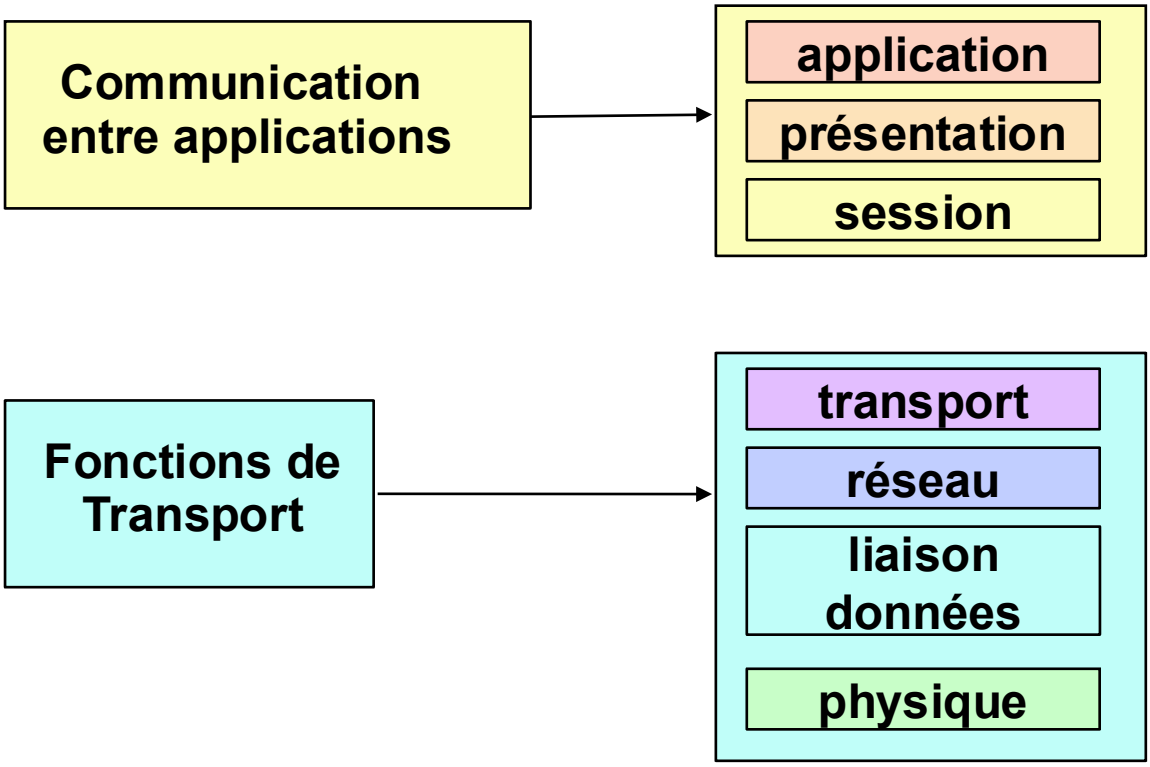
2.2.1 Le Modèle OSI

Illustration 2: Modèle OSI : Systèmes relais intermédiaires



On distingue deux types de couches : les couches hautes (applicatives) avec un haut niveau d'abstraction et les couches basses (opérées par l'infrastructure réseau) qui mettent en œuvre les mécanismes de nécessaires au fonctionnement du réseau.

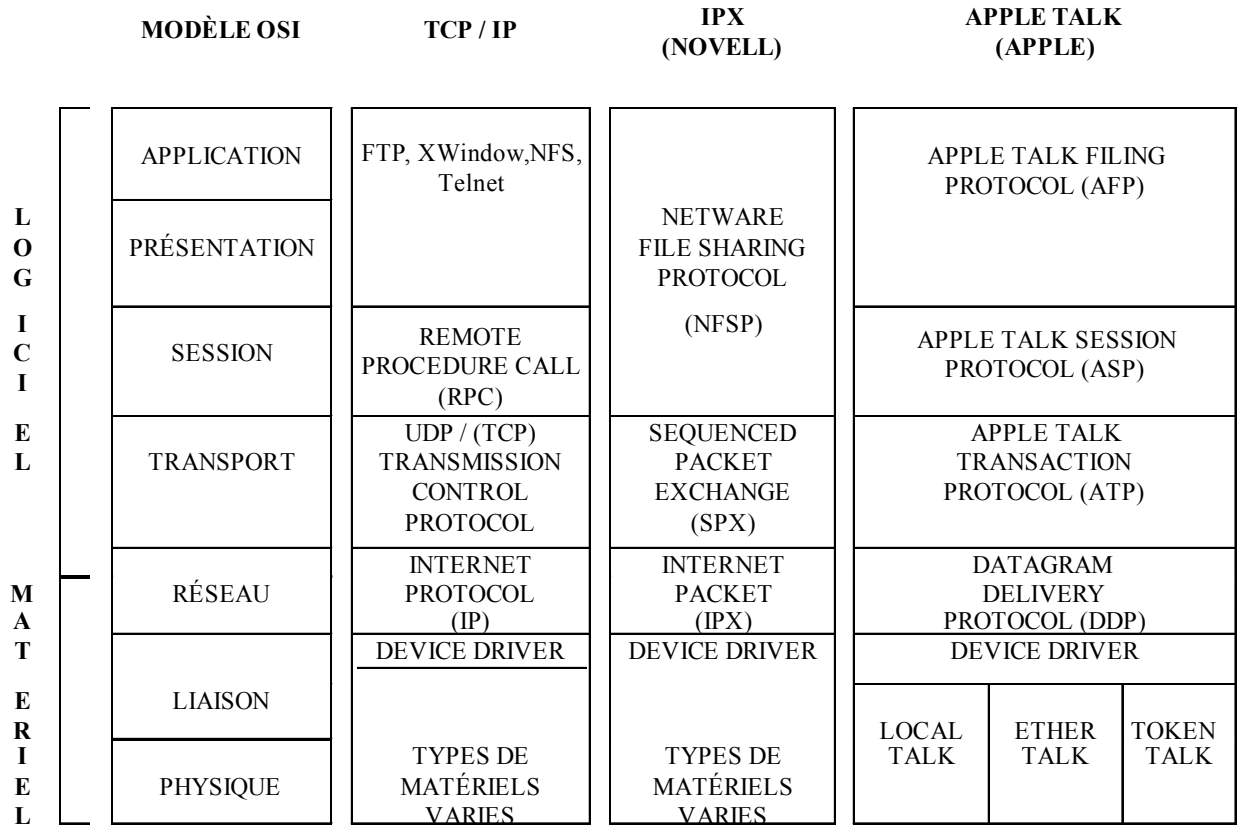
Illustration 3: Modèle OSI : Couches hautes et couches basses



Description des couches de l'OSI :

1. La couche « **physique** » est chargée de la transmission effective des signaux entre les interlocuteurs. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bit continu (notamment pour les supports synchrones).
2. La couche « **liaison de données** » gère les communications entre 2 machines adjacentes, directement reliées entre elles par un support physique.
3. La couche « **réseau** » gère les communications de bout en bout, généralement entre machines : routage et adressage des paquets.(cf. note ci-dessous).
4. La couche « **transport** » gère les communications de bout en bout entre processus (programmes en cours d'exécution).
5. La couche « **session** » gère la synchronisation des échanges et les «transactions», permet l'ouverture et la fermeture de session.
6. La couche « **présentation** » est chargée du codage des données applicatives, précisément de la conversion entre données manipulées au niveau applicatif et chaînes d'octets effectivement transmises.
7. La couche « **application** » est le point d'accès aux services réseaux, elle n'a pas de service propre spécifique et entrant dans la portée de la norme.

Illustration 4: Comparaison du modèle théorique OSI et des principaux protocoles utilisés dans l'industrie informatique



2.2.2 ICANN

Internet Corporation for Assigned Names and Numbers : *ICANN* est l'entité qui gère les adresses IP, l 'organisation de protocoles, les noms de domaines, et les serveurs « *root* » au niveau mondial.

La majorité de ces tâches étaient sous la responsabilité du gouvernement des U.S.A

- 3 Supporting Organizations (*S0*) : Recommandations sur la politique d 'Internet et sa structure :
  - *Address Supporting Organization* (*ASO*) : concerné par l 'adressage IP
  - *Domain Name Supporting Organization* : (*DNSO*) : concerné par la gestion du système de noms (*DNS*) : *gTLD*, *ccTLD*, registrars
  - *Protocol Supporting Organization* (*PSO*) : concerné par l 'affectation des paramètres des protocoles Internet
- Le *GIP Renater* est actif dans *ICANN*.
- Le Nic Français (*afnic*) est actif dans le *DNSO* de l '*ICANN*

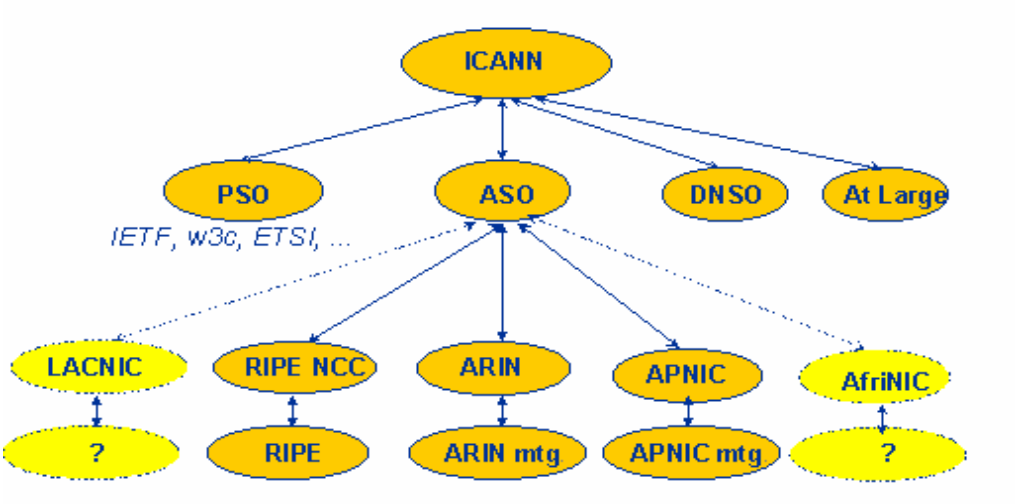


Illustration 5: Organisation de l'ICANN

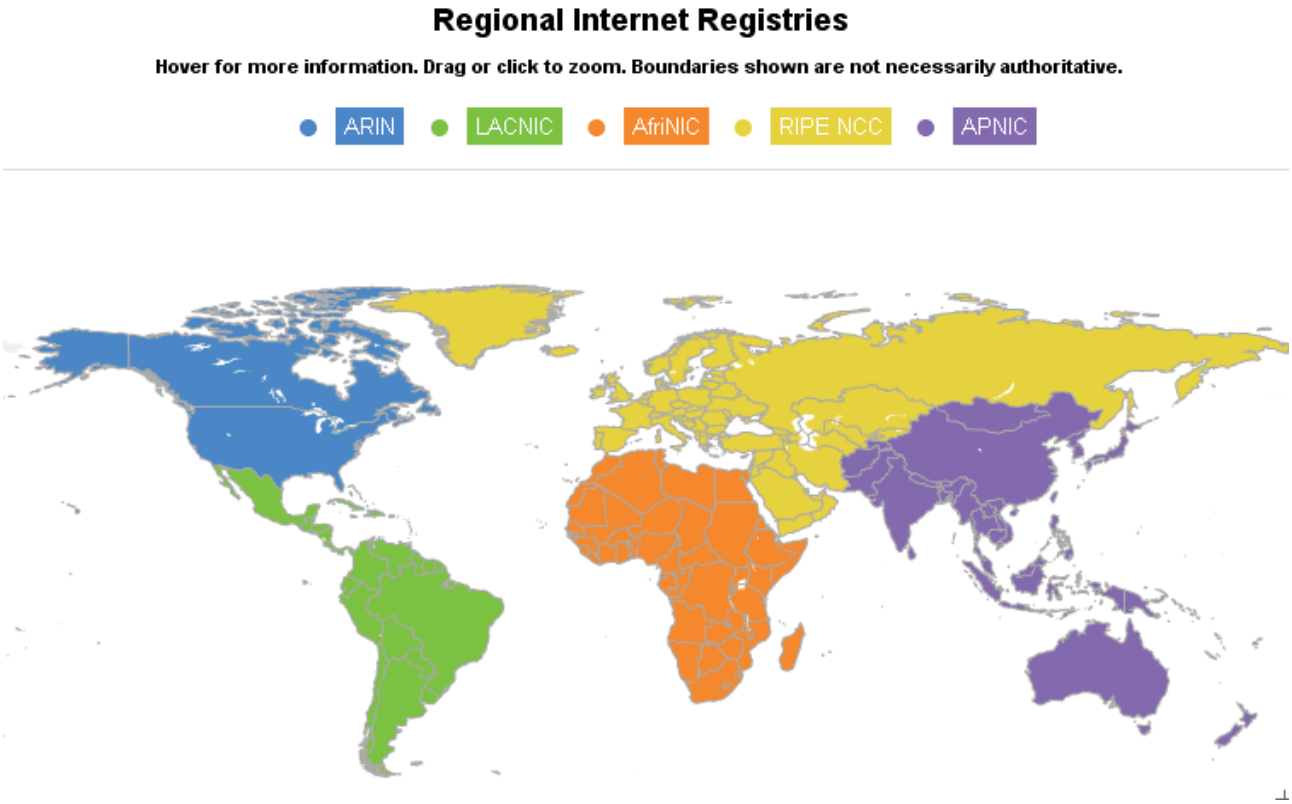


Illustration 6: Répartition mondiale des registeries Internet

Les 5 « Regional Internet Registries »

- *RIPE NCC* (*Réseaux IP Européens Network Coordination Centre*) : Europe Est/Ouest, Afrique du Nord
- *ARIN* (*American Registry for Internet Numbers*) : Amérique Nord/Sud, Afrique du Sud
- *APNIC* (*Asia Pacific Network Information Centre*) : Asie
- *LACNIC* (*Latin America and Caribbean Network Information Centre*) : Amérique du Sud et Amérique Centrale
- *AfrNIC* (*Africa Network Information Centre*) : Afrique

Le *RIPE* est l'organisme qui gère les adresses IP au plan européen. Il est un des 5 *RIR* :

- Activité (au plan européen) :
- allocation adresses IP
- allocation « *interdomain routing identifiers* » (n° *AS BGP*)
- allocation reverse *DNS* (in-addr.arpa et ip6.int)
- gestion base Whois *RIPE*
- Le GIP Renater participe aux travaux de *RIPE*.

2.2.3 IP

Les protocoles de communication *TCP/IP* sont nés à la fin des années 60, à peu près en même temps *qu'UNIX*, d'un projet du gouvernement américain pour interconnecter plusieurs sites, donc plusieurs réseaux locaux, de l'"*Advanced Research Project Agency*", d'où le nom souvent rencontré de *ARPANET*.

TCP/IP est un protocole de réseau où toutes les machines sont équivalentes, sans notion de hiérarchie et de machine maître ou centralisation. *TCP/IP* a été intégré à la version *BSD d'UNIX* dès la fin des années 70. Le succès *d'UNIX* a largement entrainé celui de *TCP/IP*.

Quelques dates majeures :

- 1963: Paul Barran de Rand définit un réseau à commutation de paquets
- 1967: Larry Roberts membre d'ARPA lance le projet ARPANET
- 1969: Vinton Cerf, 4 sites ARPANET (UCLA, UCSB, SRI, UTAH)
- 1971: Naissance de Cyclades, réseau bâtit sur les datagrammes (Louis Pouzin)
- 1973: Définition du protocole TCP à partir des idées de Cyclades
- 1974: Expérimentations entre les USA, la Gde-Bretagne et la Norvège
- 1975: Définition du protocole IP et UDP
- 1977: Ethernet sur réseau local
- 1978: IP version 4 (actuelle), mort de Cyclades (faute d'appuis)
- 1981: TCP/IP et UNIX (Université de Berkeley)
- 1982: TCP/IP intégré dans des systèmes commercialisés (Sun)
- 1983: Migration NCP -> TCP/IP sur l'ARPANET
- 1986: Initiative de la NSF pour relier ses super-calculateurs
- 1987: Premiers routeurs (Cisco, Proteon, Wellfleet)
- 1988: NSF utilise des lignes T1 à 1,5M bps, le vers de l'Internet
- 1989: Mort de l'ARPANET
- 1990: NSF utilise des lignes T3 à 45M bps
- 1991: Apparition des premiers ISP (Internet Service Providers)
- 1992: Création de l'ISOC et Naissance du WEB au CERN de Genève
- 1993: Mosaic du NCSA, Christian Huitema chairman de l'IAB
- 1994: Netscape
- 1996: Commerce électronique
- 1997: Déploiement de l'Internet domestique

Ce protocole d'interconnexion de réseau a donc largement dépassé le projet initial puisqu'il est la base de ce que l'on appelle couramment "l'internet mondial" qui inter connecte plusieurs millions d'ordinateurs dans le monde. Largement diffusé car disponible souvent gratuitement, au moins dans ses parties de base, *TCP/IP* est devenu un des piliers de ce que l'on appelle "les systèmes ouverts".

Pourquoi *IP* a supplanté la concurrence ? Le modèle *OSI* avait de nombreuses faiblesses :
- Trop de couches dans le modèle *OSI* avec une forte influence de *SNA d'IBM*
- Complexité dans la description
- Difficile à implémenter et peu efficace
- Redondance des mécanismes : contrôle d'erreur, contrôle de flux, etc. à chaque couche ou presque
- Mauvaise prise en compte des modèles sans connexion et des réseaux locaux (apparition d'Ethernet)

Le modèle *TCP/IP* s'est imposé au détriment du modèle *OSI*, mais la notion de couches ou niveaux est restée comme vision structurante des fonctions réseaux.

IP est un protocole d'acheminement de paquets entre deux extrémités (interfaces de machines) du réseau :
- il est robuste,
- efficace,
- non fiable : la perte de paquets est même nécessaire pour éliminer les boucles (*TTL*<64),
- ne prend pas en compte la notion de service (N° de port), de session...
- c'est le rôle de protocoles au dessous *d'IP* comme *UDP, TCP, SCTP*…

La robustesse du protocole IP vient de la technique de routage qui est :
- distribuée : chaque routeur prend localement une décision de routage qui est par nature, bien adaptée au contexte local
- dynamique : les tables de routage évoluent automatiquement pour tenir compte des évolutions de la topologie du réseau (perte/ajout de liens ou de routeurs) fonction de l'adresse destination : un seul paramètre pour prendre la décision de routage.

L'efficacité du protocole IP (pour les aspects débit) peut se mesurer par le ratio entre le volume de données nécessaire au protocole (en-tête, options…) et le volume de données utiles.

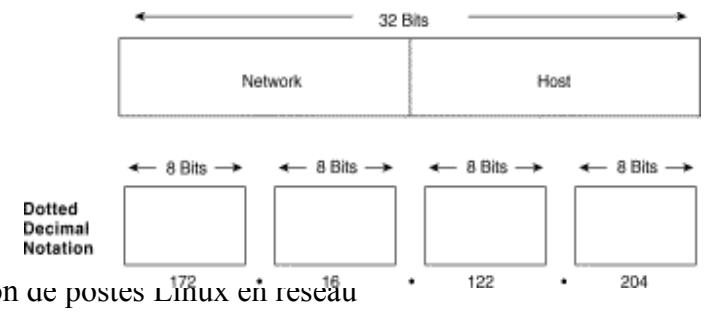
Exemple : en se plaçant dans le contexte de transport Ethernet, qui autorise une taille de paquet IP de 1500 octets au maximum, le volume d'information utilisé par le protocole IP est de 1,33% (en tête de 20 octets) en IPv4 et 2,67% (en tête de 40 octets) en Ipv6.

Ces chiffres sont comparables aux 2,47% correspondant au protocole Ethernet.

2.2.4 L'adressage IP

2.2.4.1 IPv4

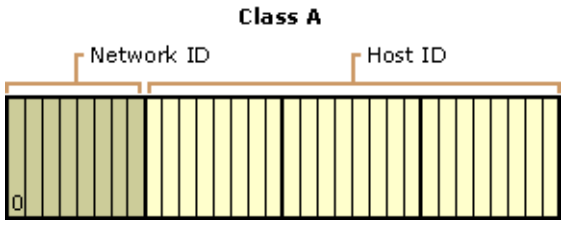
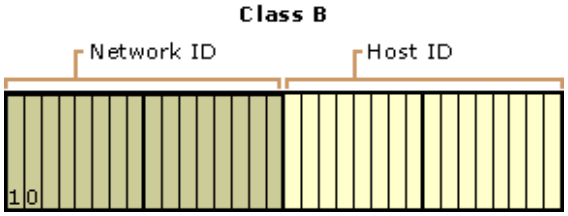
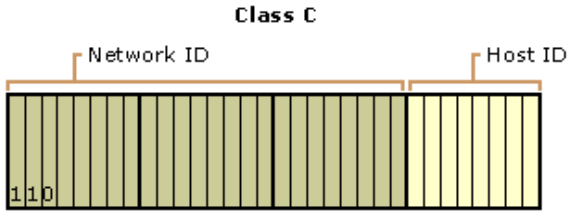
IPv4 : 32 bits (4 octets), 4 chiffres de 0 à 255 séparés par un point
Une adresse IP publique est unique au monde.



Découpée en deux :
- adresse de réseau, ou *network id :* assigné par une autorité, identifie le réseau
- identificateur local de machine, ou *host id :* assigné par l 'administrateur du réseau, identifie la machine sur le réseau

le découpage précis dépend de la classe d 'adresses…

http://www.iana.org/assignments/ipv4-address-space (allocation @ip)

	Adressage IP classe A – 7 bits pour le numéro de réseau – adresses de 1.0.0.0 à 126.0.0.0 – 24 bits pour l'adressage local – 2^{24}-2 @ locales possibles (16,277,214) – En France, pas de réseau de classe A – Ex : 16.0.0.0/8 (DEC) 18.0.0.0/8 (MIT)
	Adressage IP classe B – 16 bits pour le numéro de réseau – adresses de 128.1.0.0 à 191.254.0.0 – 16 bits pour l'adressage local – 2^{16}-2 @ locales possibles (65,534) – En France, plusieurs réseaux de classe B – Ex : IN2P3 : 134.158.0.0/16
	Adressage IP classe C – 24 bits pour le numéro de réseau – 192.0.1.0 à 223.255.255.0 – 8 bits pour l'adressage local – 2^{8}-2 @ locales possibles

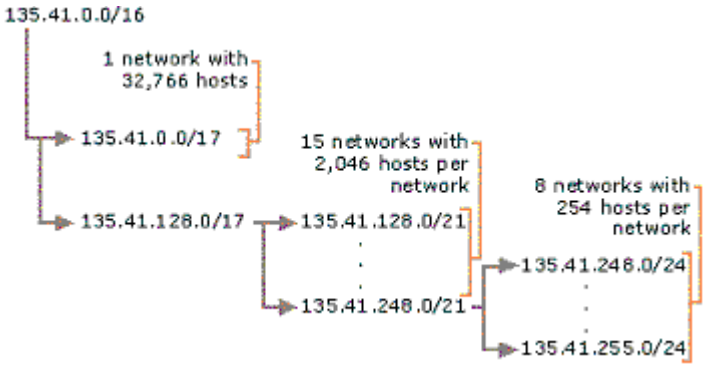
Les conventions :
- 130.190.0.0 désigne le réseau de classe B : 130.190 autrement écrit 130.190.0.0/16
- 130.190.255.255 désigne toutes les machines du réseau 130.190
  - tous les bits de la partie machine à 1 => tous les hosts du réseau
  - diffusion générale, broadcast IP

Mais on fait de moins en moins la distinction de Classes, une notion supplémentaire est apparue : les *CIDR* et les *VLSM*

Le *CIDR* (*Classless Inter Domain Routing*) abolit la notion de classe au profit d'un découpage en réseaux de taille variable. La technique de masques de sous-réseaux de taille variable (*VLSM :*

Variable Length Subnet Mask) fait son apparition. On peut alors sub-diviser un réseau en plusieurs sous réseaux, puis, pour le routage externe, regrouper ces réseaux en les agrégeant pour les annoncer dans une seule ligne dans les tables de routage des routeurs.

Exemple :



Voici dans le tableau suivant les découpages possibles des adresses Ipv4 :

IP/CIDR	Δ to last IP addr	Mask	Hosts (*)	Class	Notes
a.b.c.d/32	+0.0.0.0	255.255.255.255	1	1/256 C	
a.b.c.d/31	+0.0.0.1	255.255.255.254	2	1/128 C	d = 0 ... (2n) ... 254
a.b.c.d/30	+0.0.0.3	255.255.255.252	4	1/64 C	d = 0 ... (4n) ... 252
a.b.c.d/29	+0.0.0.7	255.255.255.248	8	1/32 C	d = 0 ... (8n) ... 248
a.b.c.d/28	+0.0.0.15	255.255.255.240	16	1/16 C	d = 0 ... (16n) ... 240
a.b.c.d/27	+0.0.0.31	255.255.255.224	32	1/8 C	d = 0 ... (32n) ... 224
a.b.c.d/26	+0.0.0.63	255.255.255.192	64	1/4 C	d = 0, 64, 128, 192
a.b.c.d/25	+0.0.0.127	255.255.255.128	128	1/2 C	d = 0, 128
a.b.c.0/24	+0.0.0.255	255.255.255.000	256	1 C	
a.b.c.0/23	+0.0.1.255	255.255.254.000	512	2 C	c = 0 ... (2n) ... 254
a.b.c.0/22	+0.0.3.255	255.255.252.000	1024	4 C	c = 0 ... (4n) ... 252
a.b.c.0/21	+0.0.7.255	255.255.248.000	2048	8 C	c = 0 ... (8n) ... 248
a.b.c.0/20	+0.0.15.255	255.255.240.000	4096	16 C	c = 0 ... (16n) ... 240
a.b.c.0/19	+0.0.31.255	255.255.224.000	8192	32 C	c = 0 ... (32n) ... 224
a.b.c.0/18	+0.0.63.255	255.255.192.000	16384	64 C	c = 0, 64, 128, 192
a.b.c.0/17	+0.0.127.255	255.255.128.000	32768	128 C	c = 0, 128
a.b.0.0/16	+0.0.255.255	255.255.000.000	65,54	256 C = 1 B	
a.b.0.0/15	+0.1.255.255	255.254.000.000	131072	2 B	b = 0 ... (2n) ... 254
a.b.0.0/14	+0.3.255.255	255.252.000.000	262144	4 B	b = 0 ... (4n) ... 252
a.b.0.0/13	+0.7.255.255	255.248.000.000	524288	8 B	b = 0 ... (8n) ... 248
a.b.0.0/12	+0.15.255.255	255.240.000.000	1048576	16 B	b = 0 ... (16n) ... 240
a.b.0.0/11	+0.31.255.255	255.224.000.000	2097152	32 B	b = 0 ... (32n) ... 224
a.b.0.0/10	+0.63.255.255	255.192.000.000	4194304	64 B	b = 0, 64, 128, 192
a.b.0.0/9	+0.127.255.255	255.128.000.000	8388608	128 B	b = 0, 128
a.0.0.0/8	+0.255.255.255	255.000.000.000	16777216	256 B = 1 A	
a.0.0.0/7	+1.255.255.255	254.000.000.000	33554432	2 A	a = 0 ... (2n) ... 254
a.0.0.0/6	+3.255.255.255	252.000.000.000	67108864	4 A	a = 0 ... (4n) ... 252
a.0.0.0/5	+7.255.255.255	248.000.000.000	134217728	8 A	a = 0 ... (8n) ... 248
a.0.0.0/4	+15.255.255.255	240.000.000.000	268435456	16 A	a = 0 ... (16n) ... 240
a.0.0.0/3	+31.255.255.255	224.000.000.000	536870912	32 A	a = 0 ... (32n) ... 224
a.0.0.0/2	+63.255.255.255	192.000.000.000	1073741824	64 A	a = 0, 64, 128, 192
a.0.0.0/1	+127.255.255.255	128.000.000.000	2147483648	128 A	a = 0, 128
0.0.0.0/0	+255.255.255.255	000.000.000.000	4294967296	256 A	

Illustration 7: Découpage IP en sous réseaux

2.2.4.2 Masque de sous-réseau

A partir d'un réseau, par exemple de classe B, on peut décider de constituer un sous-réseau comme étant un ensemble de machines qui seraient connectées sur le même réseau physique. On va pour cela fixer les adresses des machines de ce sous-réseau dans une plage bien délimitée, par exemple comme une série de classes C contigües. On va associer à ces machines un masque de sous-réseau qui délimitera de façon logique ce sous-réseau. Ce masque de sous-réseau à la même forme qu'une adresse IP. Il est construit de telle sorte qu'un 'ET' logique entre lui-même et toute adresse du sous-réseau donne la même valeur, en l'occurrence l'adresse de base du sous-réseau. Un 'ET' avec toute autre adresse donne un résultat différent.

Pour le calculer, on met à 1 tous les bits des parties réseau et sous-réseau. Ainsi dans le cas d'un sous-réseau constitue des adresses IP `134.158.136.0` à `134.158.143.255` (sous-réseau du réseau `134.158`), le masque de sous-réseau est `255.255.248.0`. Un ET entre ce masque et toute adresse du sous-réseau donne `134.158.136.0`. Un ET avec n'importe quelle autre adresse donne un résultat différent :

```
   134.158.141.10 10000110 10011110 10001101 00001010
ET 255.255.248.0  11111111 11111111 11111000 00000000
 = 134.158.136.0  10000110 10011110 10001000 00000000
```

mais

```
   134.158.144.132 10000110 10011110 10010000 10000100
ET 255.255.248.0   11111111 11111111 11111000 00000000
=  134.158.144.0   10000110 10011110 10010000 00000000
```

n'est pas dans le même réseau IP.

2.2.4.3 Autres adresses IP particulières

Un document, la RFC 1918, spécifie que l'on peut utiliser certaines adresses IP pour un réseau. Ce document fixe une convention entre tous les acteurs de l'Internet qui spécifie que certaines adresses ne seront pas routées sur Internet. Par conséquent, chaque acteur aura la possibilité d'utiliser ces adresses à l'intérieur de ce réseau sans conflit d'adressage avec les autres réseaux.

- **RFC 1918** : Address allocation for private networks
  - 10.0.0.0 - 10.255.255.255 (10/8 préfixe)
  - 172.16.0.0 - 172.31.255.255 (172.16/12 préfixe)
  - 192.168.0.0 - 192.168.255.255 (192.168/16 préfixe)
  - Utilisées pour :
    - un adressage interne à une organisation (non connectée à Internet)
    - Connectée à Internet avec translation d 'adresse (NAT ou PAT) sur une adresse IP publique d'interconnexion

- 0.0.0.0 : une machine ne connaît pas son adresse
  - station sans disque qui utilise RARP
  - client DHCP
- soi-même : 127.0.0.1 (loopback ou localhost)
- Classe D : 224.0.0.0 à 231.0.0.0
  - Adresses multicast (RFC 1700)
  - Transmissions point à multipoint; Exemple vidéo-conférence

- Pas de structuration : utilisée de façon très spéciale, ponctuelle, sans contrainte d ’unicité, sans organisation gérant leur attribution
- Adresse : Pré allocation - utilisation – libération

- Classe E :239 à 254 : Réservées pour une utilisation future

2.2.4.4 Le nommage

Cette adresse IP constituée de 4 champs numériques n'est pas conviviale pour l'utilisateur. On a donc rajouté une notion d'adresse littérale plus facilement utilisable. Cette adresse, au même titre que les champs numériques, inclue une notion de nom de machine et de nom de réseau. Cette adresse est également décomposée en champs séparés par des ".". Le premier est le nom de la machine. Les champs suivants sont le nom du domaine (réseau). Le nombre de champs n'est pas fixe pour le domaine mais est fréquemment de 2. Le dernier est toujours un champ indiquant soit le pays pour l'ensemble du monde, soit le type d'organisation pour les USA. Le champ précédent indique en général le nom de l'organisation ou société où est installé le réseau.

Pour le dernier champ (*TLD : Top Level Domain*), citons brièvement :

- **.fr** : pour la France,
- **.uk** : pour la Grande-Bretagne,
- **.de** : pour l'Allemagne

et aux USA :

- **.us**
- **.gov** pour une organisation gouvernementale;
- **.edu** pour une école ou université;
- **.mil** pour une organisation militaire;

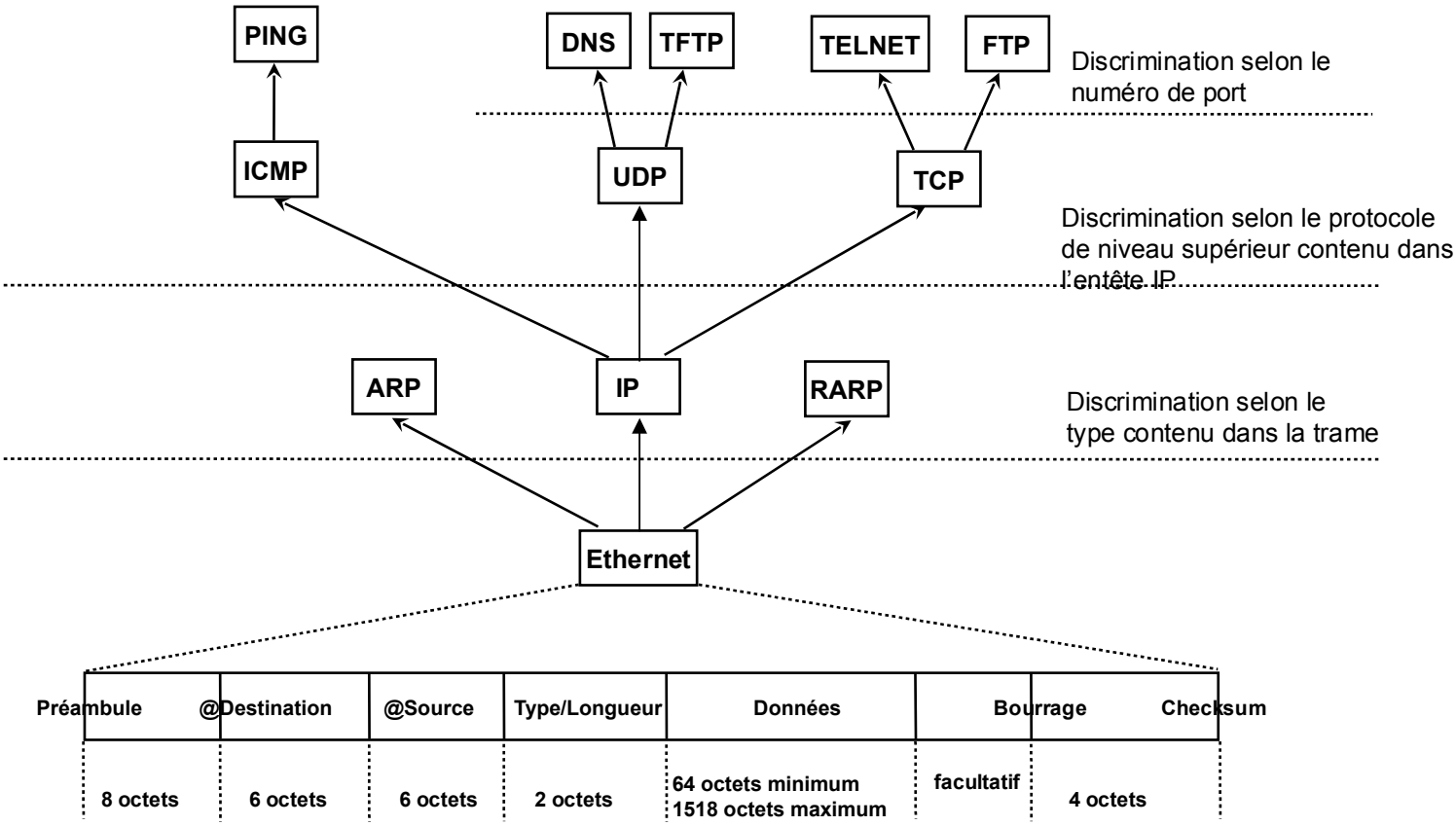
les autres TLD internationaux :

- **.int** : pour les organismes internationaux
- **.eu** : réservé aux européens
- **.name .info .biz .com .net .org** : pour toute personne ou société
- **.asia** : pour les organismes Asiatiques

Il faut noter que certaines adresses littérales apportent une notion de pays, donc de localisation du réseau, que ne contient pas l'adresse numérique.

2.2.5 Les couches de la pile IP

Illustration 8: Arborescence de la pile IP



La pile *IP* est composée de 3 couches principales :

- Réseau : *IP* :
  - *ARP*, *RARP* (*Address Resolution Protocol, Reverse Address Resolution Protocol*) : Mapping adresses IP-adresses physiques
- Transport : *TCP* et *UDP* :
  - ICMP *: Internet Control Message Protocol*
  - *TCP* : *Transmission Control Protocol*
    - mode connecté,
    - reprise automatique en cas d'erreur de transmission, de perte de paquets ou de panne de liaison entre émetteur et récepteur ,
    - identifie le flux de données comme UDP,
    - assure une transmission fiable,
    - assure un contrôle de congestion,
    - connexion uniquement en point à point *(unicast)* en mode duplex.
  - *UDP* (*User Datagram Protocol*)
    - mode déconnecté,
    - transport non fiable,
    - très efficace, (mais) peut utiliser toute la bande passante
- Application : ...

On appelle « socket » l'ensemble des informations qui procurent l'identification d'un flux de données (lié à un service) à partir du quadruplet (@IP/src, port/src, @IP/dest, port/dest).

- l'@IP/dest peut être de type multicast.

Le mode connecté du protocole TCP fonctionne de la manière suivante. Il y une phase de connexion « 3 way handshake », une phase d'échange de données et une phase de fermeture de connexion :

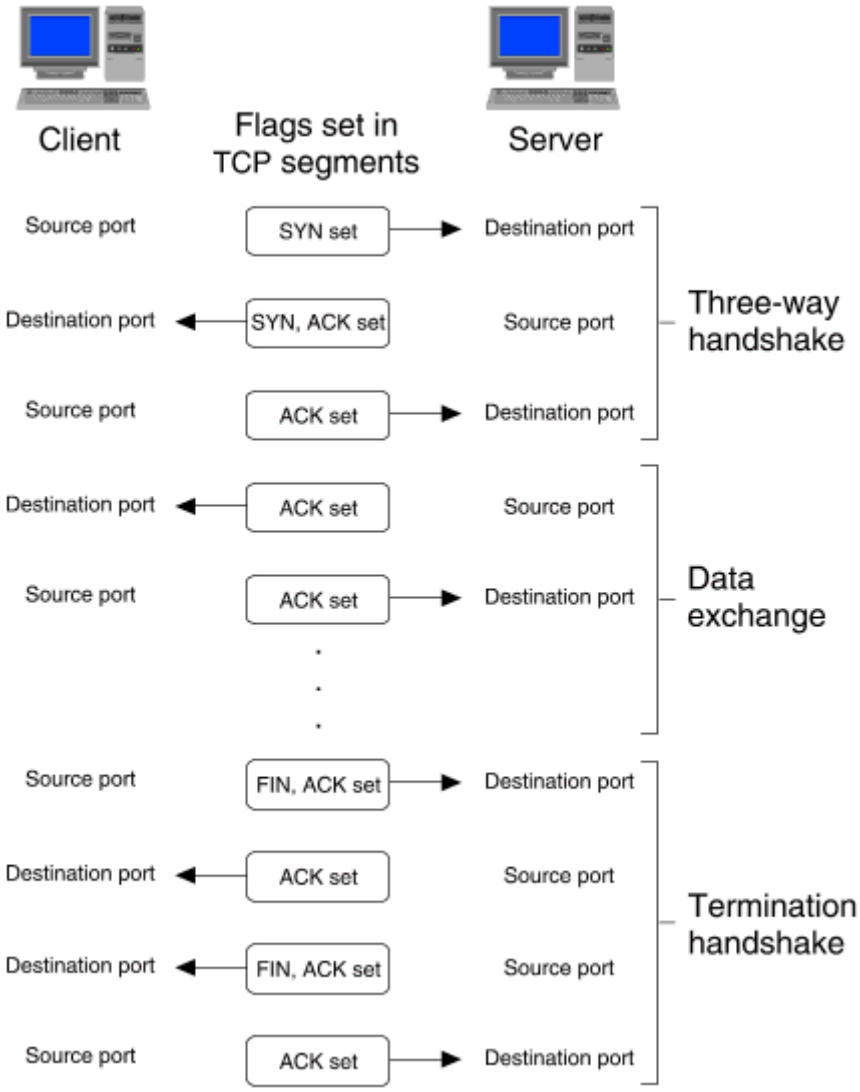


Illustration 9: Phases de connexion/déconnexion de TCP

2.2.6 Comparaison des protocoles TCP et UDP

La transmission de données par *UDP* a potentiellement de meilleures qualités isochrones que *TCP* car les paquets reçus sont immédiatement transmis à la couche applicative. C'est ce qui fait souvent préférer ce protocole pour les transmissions de données de type audio/vidéo. Cela suppose que la perte de paquets soit gérée au niveau applicatif.

Pour ne pas subir la pénalisation temporelle induite par une retransmission de données les applicatifs peuvent, soit ignorer les pertes de données quand c'est acceptable (vidéo), soit y parer en envoyant une information redondante (audio).
À contrario, la fiabilité de la transmission des données assurée par *TCP* peut se traduire, en cas de perte de paquets par un allongement du délai de transmission. En effet, il faut attendre la retransmission du paquet perdu pour que les données soient transmises à la couche applicative.
Ce délai concerne aussi tous les paquets arrivés entre temps.

Illustration 10: Liste des ports TCP et UDP

Nom	Port/protocole	Description
Echo	7/tcp et 7/udp	Echo
Discard	9/tcp et 9/udp	Discard
ftp-data	20/tcp	File Transfer protocol (data)
ftp	21/tcp	File Transfer protocol (control)
Ssh	22/tcp	Secure Shell (SSH) remote login protocol
telnet	23/tcp	Telnet
Smtp	25/tcp	Simple Mail Transfert Protocol
Time	37/tcp et 37/udp	Time
Domain	53/tcp et 53/udp	Domain Name System (DNS)
Whois	63/tcp et 63/udp	Whois++ protocol
Bootps	67/tcp et 67/udp	Bootstrap protocol server
Bootpc	68/tcp et 68/udp	Bootstrap protocol client
Tftp	69/tcp et 69/udp	Trivial File Transfer
Gopher	70/tcp et 70/udp	Gopher (ancètre du web)
Finger	79/tcp et 79/udp	Finger
http	80/tcp	World Wide Web (http)
Pop2	109/tcp	Post Office Protocol –v2
Pop3	110/tcp	Post Office Protocol –v3
Sunrpc	111/tcp et 111/udp	Sun Remote Procedure call
Ident auth	113/tcp	Authentification service
nntp	119/tcp	Network News Transfer Protocol
Netbios-ns	137/tcp et 137/udp	NETBIOS Name Service
netbios-dgm	138/tcp et 138/udp	NETBIOS Datagram Service
netbios-ssn	139/tcp et 139/udp	NETBIOS Session Service
Imap	143/tcp	Internet Message Access Protocol
xdmcp	177/tcp et 177/udp	X Display Manager Control Protocol
Bgp	179/tcp et 170/udp	Border Gateway Protocol
z39.50	210/tcp	ANSI Z39.50
Exec	512/tcp	remote process execution
Login	513/tcp	remote login a la telnet
Shell	514/tcp	Remote command
Syslog	514/udp	Log (évennements) à distance
Printer	515/udp	Spooler
Talk	517/tcp et 517/udp	Dialogue en direct

Liste non exaustive, référence http://www.iana.org/assignments/port-numbers

2.2.7 Broadcast et Multicast

S'adresser à une machine d'adresse IP bien déterminée s'appelle générer un ***unicast***. On peut avec *TCP/IP* s'adresser également à l'ensemble des machines, c'est la notion de ***broadcast***, ou s'adresser à un groupe particulier de machines, c'est la notion de ***multicast***.

Générer un *broadcast* sert, de façon générale, à demander une information à l'ensemble des machines car on ne sait pas à priori qui détient l'information. Toutes les machines traiteront cette requête comme leur étant destinée. Elles ne répondront que si elles détiennent l'information demandée. Leur réponse se fait-elle sous forme d'*unicast*. L'adresse de *broadcast* est constituée en plaçant tous les bits à 1 dans la plage variable de son sous-réseau. Cette adresse est en général, calculée automatiquement sur une machine *UNIX*, à partir du reste de la configuration,
notamment le masque de sous-réseau. Certaines vieilles versions de TCP/IP peuvent calculer l'adresse de broadcast avec tout à 0 sur la partie sous-réseau.
Ainsi dans le même exemple qu'au chapitre précédent, l'adresse de *broadcast* sera `134.158.143.255`
Les *broadcast* restent limités, en règle générale, au sous-réseau où ils sont émis (ils ne sont pas routés).

Un *multicast* est construit sur une adresse de classe D. Il sert à minimiser le trafic réseau afin d'échanger avec un groupe de machines sans pour autant générer un appel par machine. Il est utilisé par exemple pour qu'un groupe de machines suivent une vidéoconférence diffusée par une station. Les émissions de *multicast* se font donc vers des adresses IP comprises entre
`224.0.0.0` et `239.255.255.255`

Le nombre de machines à l'écoute de ces adresses *multicast* (celles qui veulent recevoir) n'est pas fixe. Les machines rejoignent ou quittent le groupe à volonté. Ce groupe peut s'étendre sur plusieurs réseaux. Les constructeurs *UNIX* ne supportent pas tous la notion de *multicast*.

2.3 Le fonctionnement d'IP

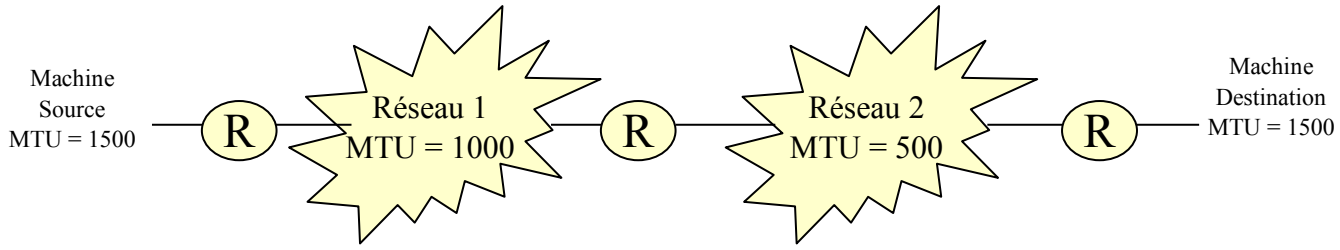
2.3.1 En émission

Le rôle de *IP* est d'encapsuler les informations à transmettre en incluant notamment les adresses *IP* source et destination. *IP* rajoute donc ce que l'on appelle l'entête IP. Cette entête, outre les 2 adresses contient une somme de contrôle, une durée de vie pendant laquelle le paquet doit arriver au destinataire, la version de *IP*, le protocole de niveau supérieur qui a fait appel à *IP*.... Les informations ainsi encapsulées par *IP* s'appellent des ***datagrammes IP***.

IP détermine notamment, grâce au masque de sous-réseau, si la machine à atteindre est sur le même réseau que l'émettrice. Il vérifie pour cela que l'application du masque de sous-réseau donne le même résultat sur les deux adresses. Dans ce cas l'adresse destinataire est directement l'adresse de la machine à atteindre. Sinon *IP* détermine l'adresse destinataire comme la prochaine machine a atteindre qui sera un relais vers la machine finale. C'est toute la notion du routage IP que nous détaillerons brièvement plus loin.

IP calcule également si la quantité d'information à transmettre est compatible avec le ***MTU***

(*Maximum Transmission Unit*) du réseau physique utilisé. Ce *MTU* correspond à la longueur maximum en octet que peut avoir une trame par rapport au débit réel du réseau (ex : 1500 pour ethernet, 4352 pour *FDDI*). Si les informations à transmettre dépassent ce *MTU*, *IP* réalise de la fragmentation pour se ramener à ce *MTU*. Si les trames doivent transiter sur plusieurs réseaux de types différents c'est le *MTU* le plus faible qui est pris. Ces fragments sont émis indépendamment les uns des autres.



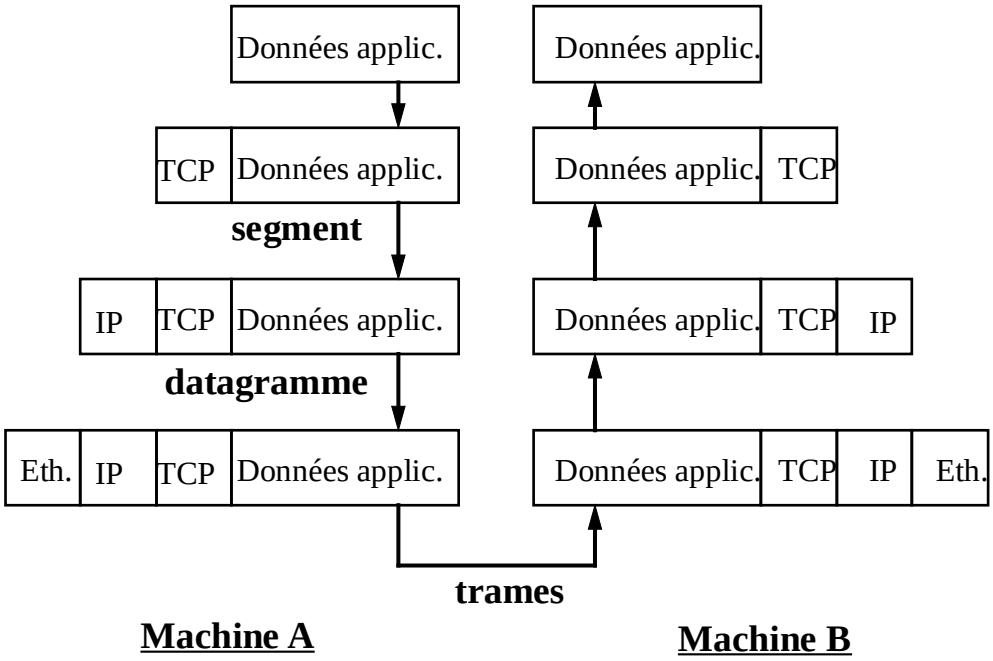
Dessin 1: Fragmentation / défragmentation ethernet

2.3.2 En réception

IP rassemble les différents datagrammes fragmentés.

IP fonctionne en mode dit *non connecté* :
Les datagrammes n'arrivent pas forcément dans le bon ordre et IP ne maintient aucune information sur leur état. IP effectue simplement une somme de contrôle sur l'entête. Si cette somme est mauvaise, le datagramme est détruit. De même IP n'effectue pas de contrôle sur les données elles-même. Dans les 2 cas, données erronées ou mauvaise entête, c'est au protocole de niveau supérieur de faire la demande de retransmission.

Au dessous de IP se situe la couche liaison (réseau physique). L'adresse IP est une adresse logique mais n'est pas l'adresse physique des cartes d'interfaces réseaux. Pour émettre, IP génère une requête d'**ARP** (Address Resolution Protocol) pour récupérer l'adresse physique correspondant à l'adresse internet du destinataire. Nous détaillerons ci-après ce mécanisme d'ARP.
On peut schématiser le cheminement d'une information depuis une application sur une machine vers l'application correspondante vers une seconde machine comme suit : (ici dans le cas d'ethernet comme liaison et TCP comme transport)



3 Les distributions Linux

Les distributions Linux trouvent leur origine dans la grande famille des systèmes *Unix*. Les distributions Linux ainsi que les systèmes *NetBSD*, *OpenBSD* et *FreeBSD* n'existeraient pas sans le travail de la communauté de développeurs Open Source qui ont (re-)développé une à une chaque brique essentielle de ces systèmes d'exploitation.

Voici un historique de la famille de systèmes *UNIX* : On voit sur ce schéma que les distributions Linux ne sont qu'une des branche de la famille des systèmes *UNIX*.

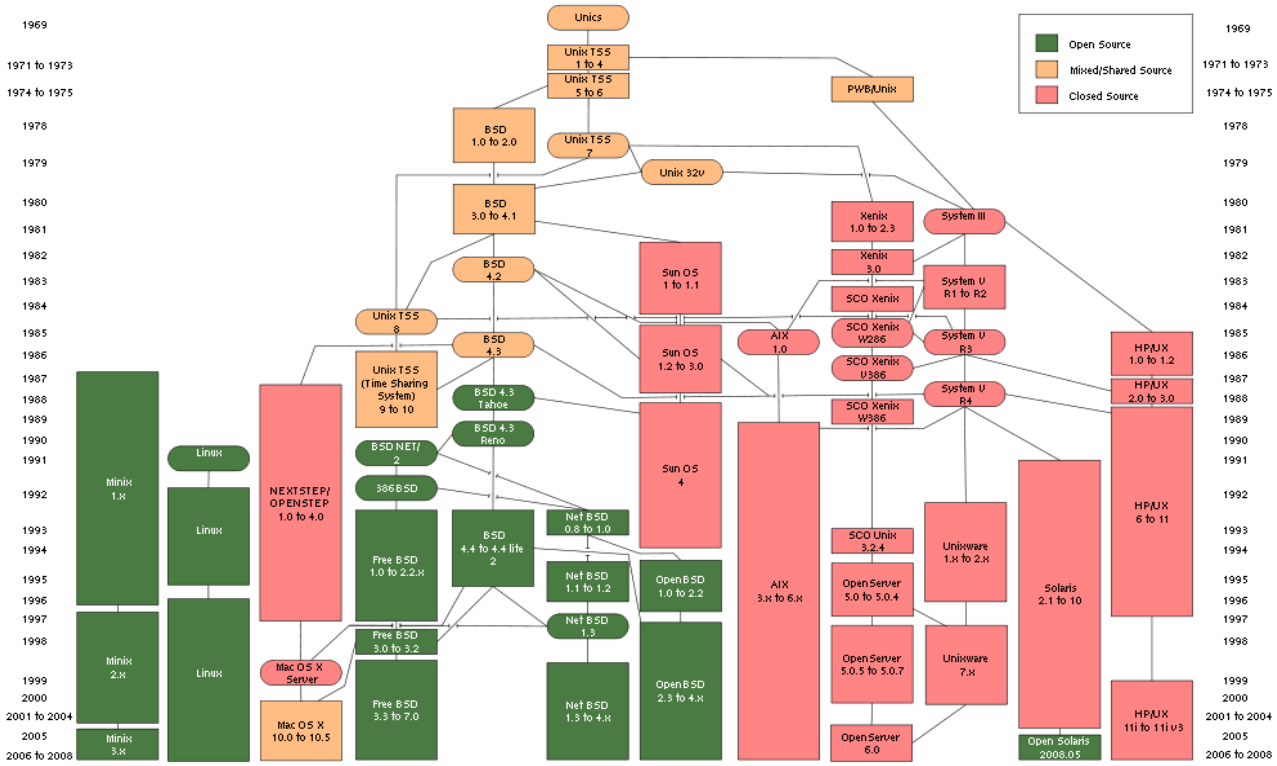


Illustration 11: Historique de la famille de systèmes UNIX

Une distribution Linux est constituée d'un assemblage de composants provenant de différentes sources. On y trouve par exemple :

- le kernel provient de http://www.kernel.org
- grub (le gestionnaire de démarrage) provient de http://www.gnu.org/software/grub
- les utilitaires cp, mv, cat... proviennent de ftp://alpha.gnu.org/gnu/coreutils
- la couche graphique (serveur X Window) xorg provient de http://xorg.freedesktop.org
- les gestionnaires de fenêtres KDE et GNOME proviennent respectivement de http://www.kde.org et http://www.gnome.org
- les clients et serveurs ssh (openssh) proviennent de http://www.openssh.com
- le shell, comme le bash provient de http://www.gnu.org/software/bash
- ...

Comme l'ensemble des sources et des scripts sont disponibles librement (principe des logiciels libres), il est possible de personnaliser la distribution, de la modifier et également de la redistribuer sous d'autres formes. C'est grâce à cette liberté que certaines distributions ont été réutilisées et modifiées et puis publiées sous la forme d'une autre distribution dérivée de la première.

Ces composants constituent un ensemble cohérent et stable dont l'installation, l'utilisation et la maintenance sont facilitées. Elles comprennent donc le plus souvent un logiciel d'installation et des outils de configuration. Il existe de nombreuses distributions, chacune ayant ses particularités. Certaines sont dédiées à un usage spécifique (pare-feu, routeur, grappe de calcul, édition multimédia…), d'autres à un matériel spécifique.

Les distributions généralistes destinées au grand public pour un usage en poste de travail

(bureautique) sont les plus connues (*Debian*, *Gentoo*, *Mandriva Linux*, *Red Hat*/*Fedora*, *Slackware, SUSE Linux Enterprise*/*openSUSE*, *Ubuntu*). Le mainteneur de la distribution peut être une entreprise (comme dans le cas de *Mandriva*, *RedHat* et *Ubuntu*) ou une communauté (comme *Debian*, *Gentoo* ou *Slackware*).

Il existe maintenant un grand nombre de distributions disponibles. Voici un exemple d'héritage (incomplet) des principales distributions Linux :

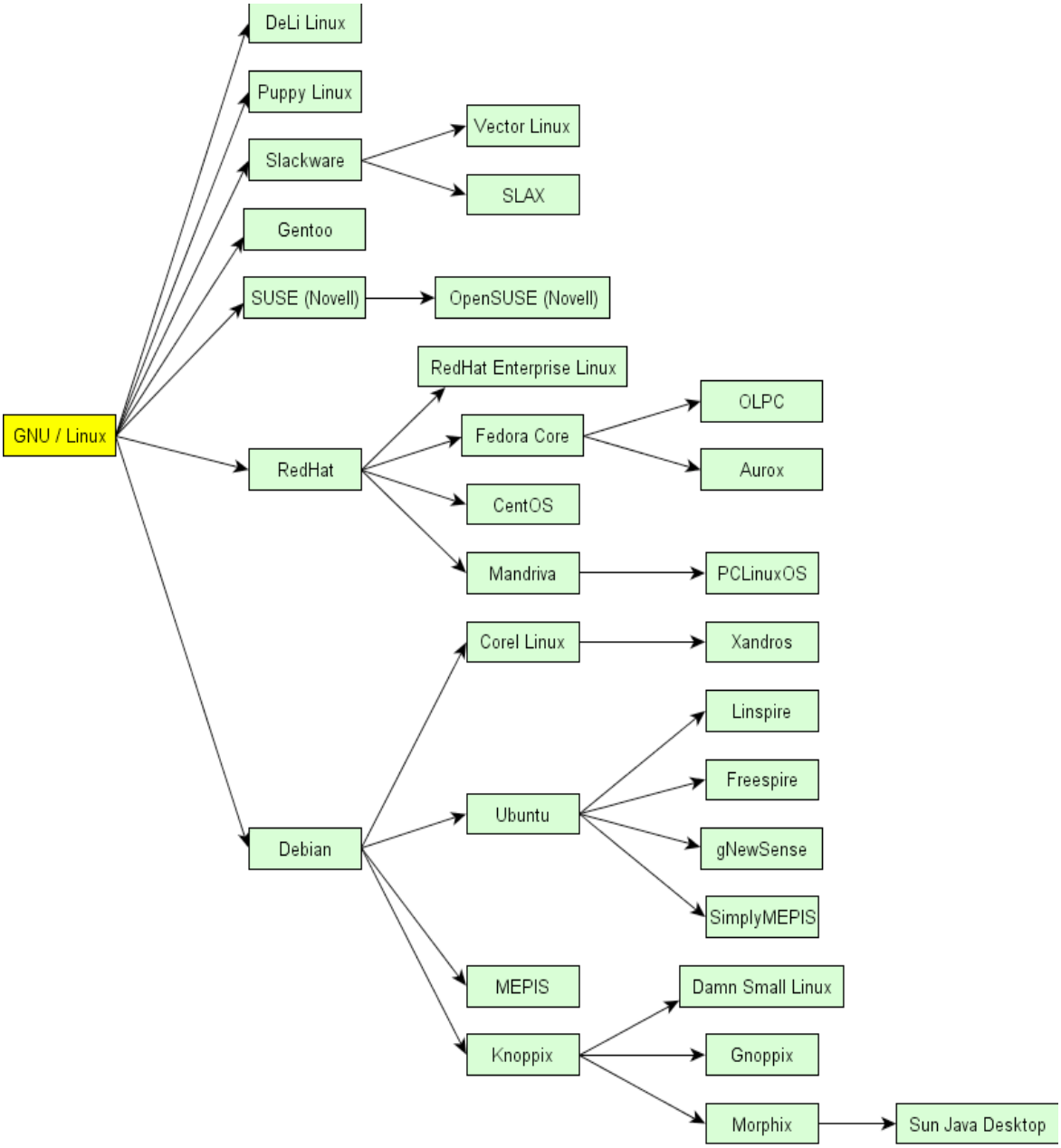


Illustration 12: Héritage des distributions Linux

3.1 Les paquetages

Une des tâches centrales de l'éditeur d'une distribution *GNU/Linux* consiste à centraliser un (plus ou moins) grand nombre de logiciels tiers, à les assembler de manière cohérente et à les empaqueter de manière à ce que les utilisateurs de la distribution puissent les installer sans difficulté.

Un paquet contient un ensemble de fichiers (utilitaires, répertoires, fichiers de configuration, scripts, documentation, fichiers source...) nécessaires au fonctionnement du logiciel à installer sur le système.

Plusieurs formats de paquets existent :

- *.tgz* (ou *.tar.gz*) : archive *tar* compressée incluant des fichiers, utilisé notamment par *Slackware*. Un paquet *tgz* comporte juste chaque fichier et le nom du répertoire où il doit être placé : pas de gestion de l'existant (versions) ni des dépendances. C'est généralement sous ce format que les développeurs mettent à disposition leurs logiciels,
- *deb*: système de gestion de paquets créé par la communauté *Debian* pour *Debian GNU/Linux* et utilisé par de très nombreuses distributions tel que *Ubuntu* ainsi que d'autres dérivées.
- *rpm* : système de gestion de paquets inventé par *Red Hat* et utilisé par *Fedora*, *SuSE*, *Mandriva* et quelques autres.
- *ebuild* : système de *Gentoo*.

3.2 Les dépôts (repositories) de paquetages

Les éditeurs de distributions *Linux* (ou les communautés dans le cas de *Debian*) mettent à disposition des utilisateurs des dépôts (généralement appelés repositories) généralement accessibles sur internet. Ces dépôts contiennent tous les paquetages installables sur la distribution. Ces dépôts sont généralement dupliqués (miroirs) à différents endroits sur Internet, dans différents pays. Ainsi, il est possible de sélectionner et d'utiliser un miroir au plus près de l'endroit où l'on se trouve.

Mais il existe aussi d'autres dépôts qui ne sont pas maintenus par les éditeurs, mais par des personnes ou entreprises qui assemblent et mettent à disposition des paquetages complémentaires non présents dans les dépôts des éditeurs.

Par exemple, pour les dépots de paquetages *RPM* pour les distributions *Red Hat*, *CentOS* et *Fedora* :

- repository *epel* : http://fedoraproject.org/wiki/EPEL
- repository *dag* (de Dag Wieers): http://dag.wieers.com
- repository *rpmforge* : http://rpmrepo.org/RPMforge (qui regroupe les contributions de Dag, Dries and Matthias Saou)
- repository *rpmfusion* : http://rpmfusion.org (qui reprends tout ce qu'il y avait dans les repositories *Livna* et *Freshrpm*)
- repository *Atrpms* (de Axel Thimm) : http://atrpms.net
- repository *dries* (de Dries Verachtert) : http://dries.ulyssis.org/rpm

Installer les mêmes paquetages que ceux existant dans la distribution officielle, mais d'une version différente provenant d'un autre repository peut mener à des incompatibilités systèmes.

Il existe aussi des bases de données permettant de rechercher dans les repositories de rpm :

- celui de Fabrice Bellet http://www.rpmfind.net
- http://rpmpbone.net

3.3 Les gestionnaires de paquetages

Un système de gestion de paquets installé et souvent spécifique à la distribution permet la recherche, l'installation, la désinstallation, la gestion des dépendances et la mise à jour de ces paquets.
Voici une liste des gestionnaires de paquetages les plus courants :

- *dpkg* : installation de paquets *Debian*
- *apt-get* : frontal pour *APT* en ligne de commande (*Debian* et autres)
- *aptitude* : frontal avancé pour APT en mode texte et ligne de commande
- *synaptic* : frontal pour APT en mode graphique GTK
- *rpm* (*RedHat Package Manager*) : gestionnaire de paquetages de *RedHat*. Il est également utilisé par toutes les distributions dérivées de *RedHat*
- *rpmdrake* et *urpmi* : Gestionnaire de paquetage de la distribution *Mandrake*
- *yum* : gestionnaire de paquetages utilisé par *RedHat* et certaines de ses dérivées

3.3.1 RPM : *RedHat Package Manager*

Le gestionnaire de paquetages *rpm* permet :

- d'installer un paquetages présent en local ou distant :
  rpm -ivh bc-1.06-21.i386.rpm
  rpm -ivh ftp://ftp.in2p3.fr/pub/linux/scientific/5x/i386/SL/bc-1.06-21.i386.rpm
- d'installer ou de mettre à jour un paquetages présent en local ou distant :
  rpm -Uvh bc-1.06-21.i386.rpm
- d'enlever un paquetages déjà installé sur le système :
  rpm -e bc
- de lister la totalité des paquetages installés sur le système :
  rpm -qa
- de savoir à quel paquetage appartient un fichier :
  rpm -qf /usr/bin/bc
- de lister les fichiers installés sur par un paquetage :
  rpm -ql bc
- d'afficher les informations sur un paquetage :
  rpm -qi bc

Dans la liste des paramètres disponibles pour la commande « *rpm -q ...* », on a :

- *--changelog* : fournit l'historique des changements de ce paquetage
- *—requires* : ce que demande ce paquetage pour pouvoir être installé, exemple :

```
# rpm -q --requires bash
/bin/bash
/bin/sh
/bin/sh
/bin/sh
config(bash) = 3.2-24.el5
libc.so.6
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.1)
libc.so.6(GLIBC_2.2)
libc.so.6(GLIBC_2.3)
libc.so.6(GLIBC_2.3.4)
libc.so.6(GLIBC_2.4)
libdl.so.2
libdl.so.2(GLIBC_2.0)
libdl.so.2(GLIBC_2.1)
libtermcap.so.2
mktemp
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rtld(GNU_HASH)
```

- *—whatrequires* : quels spont les autres paquetages qui ont besoin de ce paquetage pour pouvoir être installés, exemple :

```
# rpm -q --whatrequires bash
info-4.8-14.el5.i386
gpm-1.20.1-74.1.i386
sendmail-8.13.8-2.el5.i386
pvm-3.4.5-7.fc6.1.i386
mysql-5.0.45-7.el5.i386
sysklogd-1.4.1-44.el5.i386
dkms-2.0.17.4-1.9.noarch
vnc-server-4.1.2-14.el5_3.1.i386
ypbind-1.19-11.el5.i386
rsyslog-2.0.6-1.el5.i386
initscripts-8.45.25-1.el5.i386
vixie-cron-4.1-76.el5.i386
squid-2.6.STABLE21-3.el5.i386
```

Bien qu'il gère les dépendances (il refuse d'installer un paquetage si les dépendances de ce paquetage ne sont pas résolues, il refuse également de désinstaller un paquetage si un autre paquetage a besoin de lui) le gestionnaire de paquetages *rpm* n'installe (ou ne désinstalle) pas automatiquement les dépendances. C'est pour cela qu'il est utilisé généralement pour la gestion individuelle des paquetages.

Exemple d'une impossibilité d'installation du paquetage abiword à cause d'un problème de dépendance :

```
# rpm -Uvh abiword-2.4.6-6.fc8.i386.rpm
warning: abiword-2.4.6-6.fc8.i386.rpm: Header V3 DSA signature: NOKEY, key ID
4f2a6fd2
error: Failed dependencies:
        libc.so.6(GLIBC_2.7) is needed by abiword-2.4.6-6.fc8.i386
        libenchant.so.1 is needed by abiword-2.4.6-6.fc8.i386
        libfribidi.so.0 is needed by abiword-2.4.6-6.fc8.i386
        libgoffice-1.so.2 is needed by abiword-2.4.6-6.fc8.i386
        liblink-grammar.so.4 is needed by abiword-2.4.6-6.fc8.i386
        libmathview.so.0 is needed by abiword-2.4.6-6.fc8.i386
        libmathview_frontend_libxml2.so.0 is needed by abiword-2.4.6-6.fc8.i386
        libots-1.so.0 is needed by abiword-2.4.6-6.fc8.i386
        libpopt.so.0(LIBPOPT_0) is needed by abiword-2.4.6-6.fc8.i386
        link-grammar >= 4.2.2 is needed by abiword-2.4.6-6.fc8.i386
        mathml-fonts is needed by abiword-2.4.6-6.fc8.i386
```

Alors que la mise à jour et l'installation des paquetages avec beaucoup plus aisée car *yum* résout lui même les dépendances entre paquetages.

Pour une gestion automatique les dépendances, il est préférable d'utiliser *YUM*.

3.3.2 Yellow dog Updater Modified (Yum)

Créé par *Yellow Dog Linux* et utilisé principalement par les distributions *Fedora* et *Red Hat Enterprise Linux*.

Il permet de gérer l'installation et la mise à jour des logiciels installés sur une distribution *GNU/Linux*. C'est une surcouche à *RPM* gérant les téléchargements et les dépendances, de la même manière que *APT* de *Debian* ou *Urpmi* de *Mandriva*.

Les distributions sont constituées d'un ensemble d'utilitaires et de fichiers distribuées généralement sous forme de packages (.rpm, .deb, .tar.gz ...). Il devient alors plus aisé de mettre à jour un utilitaire en installant alors le package qui contient l'utilitaire corrigé.

Les fichiers de configuration de l'utilitaire de mise à jour *YUM* sont :

/etc/yum.conf : pour les paramètres généraux de l'utilitaire

/etc/yum.repos.d/* : pour les paramètres concernant les dépôts où *yum* pourra télécharger des packages

```
# cat /etc/yum.repos.d/sl5x.repo
[sl-base]
name=SL 5 base
baseurl=ftp://ftp.scientificlinux.org/linux/scientific/5x/$basearch/SL
enabled=1

# cat /etc/yum.repos.d/sl5x-security.repo
[sl-security]
name=SL 5 security updates
baseurl=ftp://ftp.scientificlinux.org/linux/scientific/5x/
$basearch/updates/security
enabled=1
```

Ici, sur un système basé à partir de packages .rpm, on peut utiliser les commandes *rpm* et *yum* :

```
# rpm -qa |grep server
xorg-x11-server-Xvfb-1.1.1-48.26.el5_1.5.i386
xorg-x11-server-sdk-1.1.1-48.26.el5_1.5.i386
xorg-x11-server-utils-7.1-4.fc6.i386
VMware-server-1.0.4-56528.i386
openssh-server-4.3p2-24.el5.i386
evolution-data-server-1.8.0-25.el5.i386
setroubleshoot-server-2.0.5-3.el5.noarch
evolution-data-server-devel-1.8.0-25.el5.i386
vnc-server-4.1.2-9.el5.i386
xorg-x11-server-Xorg-1.1.1-48.26.el5_1.5.i386
xorg-x11-server-Xnest-1.1.1-48.26.el5_1.5.i386
tomcat5-server-lib-5.5.23-0jpp.3.0.3.el5_1.i386
mysql-server-5.0.45-7.el5.i386

# yum install bind
Setting up Install Process
Setting up repositories
Loading mirror speeds from cached hostfile
Reading repository metadata in from local files
Excluding Packages in global exclude list
Finished
0 packages excluded due to repository priority protections
0 packages excluded due to repository protections
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for bind to pack into transaction set.
bind-9.3.4-6.P1.el5.i386. 100% |=========================|  52 kB    00:00
---> Package bind.i386 30:9.3.4-6.P1.el5 set to be updated
--> Running transaction check
Beginning Kernel Module Plugin
Finished Kernel Module Plugin

Dependencies Resolved

================================================================================
 Package                 Arch       Version          Repository        Size
================================================================================
Installing:
 bind                    i386       30:9.3.4-6.P1.el5  sl-security       957 k

Transaction Summary
================================================================================
Install      1 Package(s)
Update       0 Package(s)
```

```
Remove       0 Package(s)

Total download size: 957 k
Is this ok [y/N]: y
Downloading Packages:
(1/1): bind-9.3.4-6.P1.el 100% |=========================| 957 kB    00:01
Running Transaction Test
warning: bind-9.3.4-6.P1.el5: Header V3 DSA signature: NOKEY, key ID 82fd17b2
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: bind                         ########################## [1/1]

Installed: bind.i386 30:9.3.4-6.P1.el5
Complete!
```

Lors des mises à jour, il faut bien vérifier que les fichiers de configuration n'ont pas été changés, on peut le remarquer par la présence de fichiers de configuration avec les extensions : exemple pour le fichier */etc/ntp.conf*

/etc/ntp.conf.rpmnew : un nouveau fichier de configuration */etc/ntp.conf.rpmnew* a été ajouté en plus du fichier de configuration */etc/ntp.conf* actuellement utilisé, des options nouvelles ont fait apparition depuis la précédente version : il faut vérifier les options pour mettre a jour le fichier de configuration */etc/ntp.conf*

/etc/ntp.conf.rpmsave : le fichier de configuration */etc/ntp.conf* a été sauvegardé en */etc/ntp.conf.rpmsave* et un nouveau fichier */etc/ntp.conf* a pris sa place.

Voir en annexe les différentes utilisations de cette commande.

4 Méthodes d'installation de distributions linux

4.1 Média et méthodes d'installation

La plupart des distributions Linux proposent plusieurs médias d'installation selon le choix de l'utilisateur. Les plus courantes sont :

- CDROM : on choisi le lecteur de *CDROM* ou *DVD-ROM*. La plupart des distributions actuelles ne rentrent plus sur un seul *CDROM*, Par exemple, la distribution Scientific-Linux demande 8 CDROM ou 2 DVD. Cette méthode nécessite de télécharger la totalité de la distribution, soit environ 8Go. Ce qui est pratique quand on veut installer Linux sur une machine ne disposant pas d'accès Internet, mais assez inefficace compte tenu de la quantité de données téléchargées par rapport à ce qui sera réellement installé.
- On peut utiliser un mini boot, sorte d'image minimale de boot dont la taille est très inférieure à la taille totale de la distribution. Généralement, cette image de taille réduite ~10Mo peut être gravée sur un CDROM ou alors installée sur une clé USB. Cette méthode d'installation nécessite d'avoir un accès Internet, ou alors d'avoir à disposition sur un réseau local un serveur FTP ou HTTP dans lequel seront accessibles l'ensemble des fichiers paquetages nécessaires à l'installation de la distribution Linux sur le disque dur de la machine.
  - Mini-boot sur CDROM : Il est nécessaire de télécharger une petite image de boot (exemple : http://ftp.in2p3.fr/linux/scientific/5x/i386/images/boot.iso : taille < 10Mo) que l'on grave sur un CDROM. Lors de l'installation, on démarrera sur ce CDROM et lors du choix de la méthode, on indiquera l'URL ou télécharger les paquetages nécessaires à l'installation de la distribution. Exemple : http://ftp.in2p3.fr/linux/scientific/5x/i386 .
  - Mini-boot sur une clé USB : Il est nécessaire de télécharger une petite image de boot

(exemple : http://ftp.in2p3.fr/linux/scientific/5x/i386/images/diskboot.img : taille ~10Mo) que l'on copie sur une clé USB à l'aide de la commande suivante (dans l'exemple, diskboot est le fichier image que l'on a téléchargé et /dev/sdb est le « device » correspondant à une clé USB sur une machine Linux) :

```
dd if=diskboot.img of=/dev/sdb
```

Lors de l'installation, on démarrera sur le CDROM ou la clé USB et lors du choix de la méthode d'installation, on indiquera l'URL (FTP ou HTTP) où télécharger les paquetages nécessaires à l'installation de la distribution. Exemple : http://ftp.in2p3.fr/linux/scientific/5x/i386 .

- disque dur : Cette méthode demande à ce que l'ensemble de l'arborescence des paquets de la distribution soient stockés sur une partition du disque dur local que la méthode de boot doit reconnaitre, il faut spécifier une partition contenant les images des *CDROM* d'installation.
- image *NFS* : il faut disposer d'un serveur *NFS* accessible qui contient toute l'arborescence d'installation qui est contenue dans les *CDROM* d'installation.
- *PXE* : *PXE* est une méthode de démarrage d'une machine par le réseau. Cette méthode consiste à faire télécharger une image *PXE* de boot depuis un serveur *tftp*, après avoir démarré, il sera possible de choisir la méthode pour télécharger les paquetages lors de l'installation.

Le *CDROM* de boot contient aussi généralement un mode appelé « *rescue* » qui permet de booter sur le *CD*, puis de monter les systèmes de fichiers situés sur les partitions du disque local pour éventuellement réparer des problèmes de disque, de configuration, de réseau...

À noter que certains *CDROM* de boot contienne aussi généralement des utilitaires de test système, comme l'utilitaire « *memtest* » qui permet de tester de manière intensive la mémoire système de manière a détecter les problèmes matériels.

4.2 Méthodes d'installations automatiques

Il existe aussi des méthodes d'installation automatiques. Ces méthodes utilisent plusieurs fonctionnalités qui une fois assemblées, permettent d'automatiser l'installation d'une ou de plusieurs machines. Les fonctionnalités généralement utilisées par les méthodes d'installation automatiques sont les suivantes :

- *boot PXE* ou boot à partir d'une mini image *CD* ou *USB*,
- choix automatique (ou pré-déterminé) de la méthode d'installation (*CDROM*, *FTP, HTTP, NFS*...)
- choix automatique (ou pré-déterminé) du partitionnement,
- choix automatique (ou pré-déterminé) des paquetages à installer,
- choix automatique (ou pré-déterminé) de la configuration,
- post-installation automatique.

Celle retenue par *Scientific Linux* (qui est dérivée de *RedHat Enterprise Linux*, comme *CentOS*) est appelée *kickstart*. *Kickstart* s'appuie sur un fichier dont la syntaxe est documentée, ce fichier contient toutes les instructions décrivant les actions que doit entreprendre l'utilitaire *kickstart* pour installer et initialiser la machine sur laquelle on installe le système Linux.

Elle fait appel à un fichier de configuration que l'on peut placer sur une disquette, une clé *USB* ou un serveur http, ftp ou *NFS*...

Voici un exemple d'un fichier kickstart :

```
# Kickstart file automatically generated by anaconda.

install
url --url ftp://ftp.in2p3.fr//pub/linux/scientific/5x/i386
lang fr_FR.UTF-8
keyboard fr-latin1
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$AH8tJfqr$D17VVaDnzFrsRpPaJhjmR21
firewall --enabled --port=22:tcp
authconfig --enableshadow --enablemd5
selinux --enforcing
timezone --utc Europe/Paris
bootloader --location=mbr --driveorder=hda --append="rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --linux --drives=hda
#part /boot --fstype ext3 --size=100 --ondisk=hda
#part pv.2 --size=0 --grow --ondisk=hda
#volgroup VolGroup00 --pesize=32768 pv.2
#logvol / --fstype ext3 --name=LogVol00 --vgname=VolGroup00 --size=1024 --grow
#logvol swap --fstype swap --name=LogVol01 --vgname=VolGroup00 --size=400 --grow
 --maxsize=800

%packages
@editors
@text-internet
@legacy-network-server
@dns-server
@gnome-desktop
@core
@base
@ftp-server
@network-server
@base-x
@graphics
@web-server
@smb-server
@printing
@mail-server
@french-support
@server-cfg
@workstation
@graphical-internet
device-mapper-multipath
vnc-server
xorg-x11-utils
xorg-x11-server-Xnest
agg
libsane-hpaio
-sysreport
```

Notez qu'un fichier *kickstart* est systématiquement généré par anaconda lors de toute nouvelle installation de la distribution linux. Il est stocké sur le compte root du système (fichier /root/anaconda-ks.cfg). On peut alors le personnaliser et s'en servir de template pour réaliser des installations automatisées.

Si on dépose ce fichier (par exemple appelé install.ks) sur le serveur web http://monlab.fr, on pourra fournir en argument d'installation lors de la séquence de boot sur le média d'installation :

linux ks=http://monlab.fr/install.ks

Ce fichier contient différentes directives décrivant :

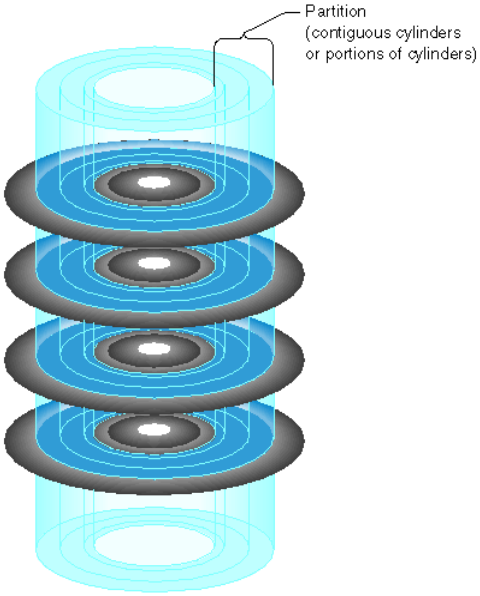
- les fonctions de langues (*lang*) de type de clavier (*keyboard*) et temps (*timezone*)

- configuration réseau (*network*)
- mot de passe root par défaut (*rootpw*)
- les options de sécurité (*firewall, authconfig, selinux*)
- la méthode de partitionnement (*bootloader, clearpart, part, volgroup, logvol*) et de formatage *(--fstype*)
- et une section (***%packages***) consacrée aux paquetages installés initialement dans le système :
  - *@editors* : spécifie l'ensemble des paquetages classés dans la catégorie *editors*)
  - *vnc-server* spécifie que le paquetage *vnc-server* doit être installé

5 Partitionnement disque et systèmes de fichiers

5.1 Partitionnement

Il existe plusieurs types de tables de partitions : *DOS*, *BSD*, *SUN*, *SGI*... La structure de la table des partitions est différente suivant le type de la table.



Sur les tables de partitions de type « *DOS* » (telles que savent gérer les *BIOS* présent sur les PC), un même disque dur peut contenir jusqu'à 4 partitions principales ou 3 partitions principales et une partition étendue. Dans une partition étendue, vous pouvez définir autant de partitions que vous le désirez. Les partitions contenues dans la partition étendue s'appellent des partitions logiques.
Les autres types de tables de partitions ont des caractéristiques différentes, par exemple, les tables de partition *SGI* ont une limite de 16 partitions par disque.

La table de partition est inscrite sur quelques octets du premier bloc présent sur le disque dur du système. Ainsi, pour sauvegarder la table de partition d'un disque, il suffit par exemple de taper la commande suivante :

```
dd if=/dev/hda of=sauvegarde-table-partitions bs=512 count=1
```

ou :
- if=/dev/hda : désigne le disque ou est situé la table des partitions (ici /dev/hda)
- of=sauvegarde-table-partitions : le fichier créé qui contiendra la table
- bs=512: block size = 512 octets
- count=1: on sauvegarde un seul bloc

Selon le même principe, effacer la table des partitions est une opération qui prends moins d'une seconde et qui n'efface en rien les données présentes sur le disque. Il est d'ailleurs possible de recouvrer une table de partitions effacée.

Pour installer un système d'exploitation, vous avez besoin d'une partition principale tandis que, pour sauvegarder des données, une partition logique suffit.

Donc, s'il vous est impossible de choisir de créer une partition logique, soit vous avez atteint la limite (4 partitions principales), soit il n'y a pas assez d'espace libre dans la partition étendue ou à côté de celle-ci.

Les principales caractéristiques d'une partitions sont les suivantes :

- est-elle bootable ? (drapeau : oui, non)
- numéro de bloc de début de la partition,
- taille de la partition,
- type de partition.

Dans la gestion des unités logiques, la seule différence entre Microsoft *Windows* et *GNU/Linux* réside dans le nommage des partitions. *Windows* utilise des lettres (d'où une certaine limite en nombre de partitions). *GNU/Linux* utilise un combiné de lettres et de numéros mais les lettres ne représentent en fait que le type de matériel (*HD* pour un disque dur interne, *SD* pour un disque *SCSI*). Exemple :

- pour un disque dur reconnu en tant que /dev/hda, la partition numéro 1 sera identifiée comme /dev/hda1
- pour un disque dur reconnu en tant que /dev/sda, la partition numéro 1 sera identifiée comme /dev/sda1

Il existe différents logiciels de gestion de partitions sous Linux : Les plus courants sont *fdisk* et *parted* (ligne de commande), ainsi que les outils graphiques *gparted* et *qtparted*.

Voici un exemple des types de partitions reconnus par l'utilitaire *fdisk* sous Linux :

```
 0  Empty           1e  Hidden W95 FAT1 75  PC/IX           be  Solaris boot
 1  FAT12           24  NEC DOS         80  Old Minix       bf  Solaris
 2  XENIX root      39  Plan 9          81  Minix / old Lin c1  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  82  Linux swap      c4  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     83  Linux           c6  DRDOS/sec (FAT-
 5  Extended        41  PPC PReP Boot   84  OS/2 hidden C:  c7  Syrinx
 6  FAT16           42  SFS             85  Linux extended  da  Non-FS data
 7  HPFS/NTFS       4d  QNX4.x          86  NTFS volume set db  CP/M / CTOS / .
 8  AIX             4e  QNX4.x 2nd part 87  NTFS volume set de  Dell Utility
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e1  DOS access
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS          e4  SpeedStor
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
10  OPUS            55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fd  Linux raid auto
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fe  LANstep
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
1c  Hidden W95 FAT3
```

Il existe également des gestionnaires de volume comme *LVM* (Red Hat) ou *EVMS*. Ces gestionnaires de volumes utilisent une partition logique ou principale comme contenant et offrent la possibilité de contenir eux même d'autres partitions.

Les gestionnaires de volume définissent 3 niveaux d'abstraction sur les unités de stockage :

- les volumes physiques (*PV : Physical Volume*)
- les volumes groupes (*VG : Volume Group*)
- les volumes logiques (*LV : Logical Volume*)

Les niveaux d'abstraction s'imbriquent de la manière suivante :

- Un *Physical Volume* est contenu dans une partition ou un disque
- Un *Volume Group* regroupe plusieurs *Physical Volume* en un seul groupe
- Les *Logical Volume* sont définis au dessus d'un *Volume Group*.

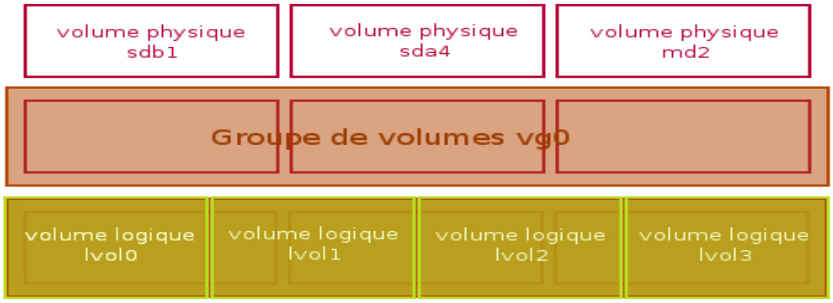
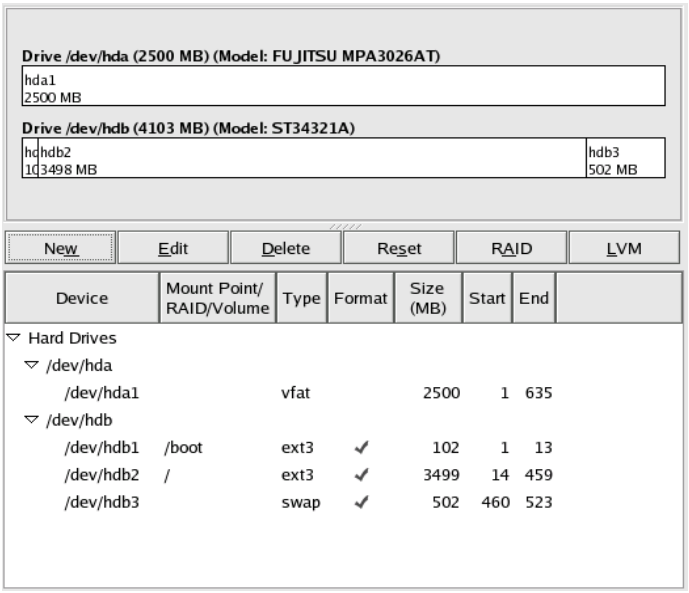


Illustration 13: Organisation des LVM

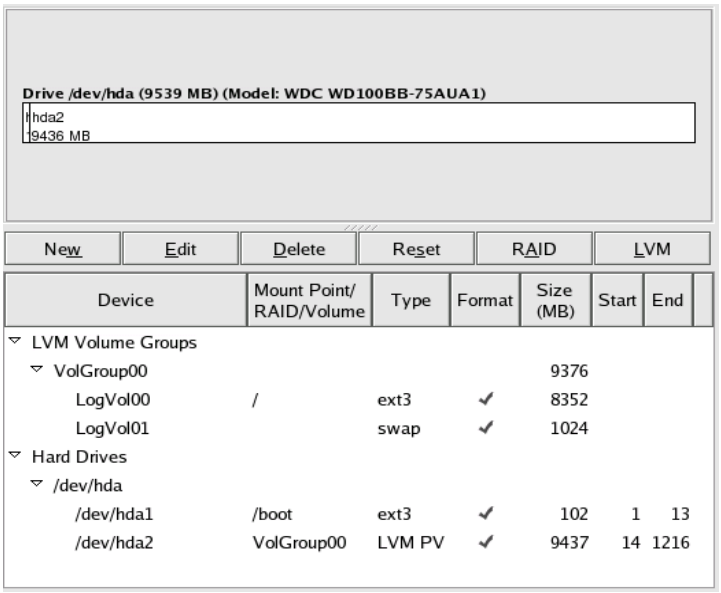
Voici quelques caractéristiques intéressantes qu'offrent les gestionnaires de volumes :

- possibilité d'agrandir des *Volume Group* en agrégeant plusieurs *Physical Volume* (par exemple sur plusieurs disques pour améliorer les performances),
- possibilité d'agrandir des *Logical Volume* en tenant compte des modifications des *Volume Group*,
- l'agrandissement ou la réduction de taille des sous-partitions contenues dans un *Logical Volume*,
- Chaque *Physical Volume* est identifié par un identifiant unique : *UUID*. De ce fait, déplacer un disque dans un système (qui a pour effet de modifier le *device* qui permet d'y accéder) permet à *LVM* de retrouver le *Physical Volume* même si le device a changé.
- Il est possible de déplacer un *Logical Volume* de *Volume Group* en cours de fonctionnement, sans interruption de service,
- possibilité de créer des snapshots (lecture seule ou lecture/écriture) sur les *Logical Volume*

Sous *Linux*, il est possible de définir des partitions natives, ou des partitions contenues dans un volume *LVM* (*Logical Volume Manager*) :



Dessin 3: Partitions natives



Dessin 2: Partitions dans un volume LVM

5.2 Systèmes de fichiers

Il existe un grand nombre de systèmes de fichiers, certains seulement sont supportés par *Linux*. Les systèmes de fichiers principalement utilisés sous Linux sont :

- *Linux Ext2* (*Ext2FS*), utilisé par le système Linux et non reconnu nativement par *MS-DOS* et tous les systèmes *Windows* (il existe toutefois des utilitaires qui ajoutent cette fonctionnalité aux systèmes sous *Windows*. La gestion des droits d'accès (utilisateurs, groupes) sont pris en compte. La taille maximum d'un système de fichiers *ext2* est de 4To, un fichier ne peut pas dépasser 2Go, ni 255 caractères dans le nom.
- *Linux Ext3* (*Ext3FS*), utilisé par le système Linux et non reconnu par *MS-DOS* et tous les systèmes *Windows*. La taille maximum d'un système de fichiers *ext3* est de 16 To. C'est une amélioration du *Ext2FS* auquel a été ajouté la journalisation des fichiers afin de permettre de rattraper rapidement les incohérences dans le système de fichiers après un démontage sauvage.
- *Linux Ext4* : Ce récent système de fichier journalisé est une évolution du système de fichiers *ext3* et reste compatible en partie avec ce dernier. La limite de capacité *d'ext4* est d'1 Exa Octets (1 million de To) en utilisant des numéros de blocs sur 48 bit et des blocs de 4Ko.
- *Reiser* (*ReiserFS*), utilisé par *Linux*, ce système est journalisé.
- *JFS* : système de fichier journalisé développé par *IBM*
- *XFS* : système de fichier journalisé développé par *SGI*
- *Linux Swap* : utilisé par le système *Linux*. Ce n'est pas un système de fichier, mais simplement une partition utile au système pou gérer le fichier d'échange de *Linux* (le *Swap*).

Il existe aussi des systèmes de fichiers spécifiquement conçus pour des clusters Ces systèmes de fichiers sont dits « partagés », c'est à dire qu'il permettent l'accès aux mêmes données par plusieurs serveurs. Ces systèmes de fichiers supportent les accès concurrents, le verrouillage décentralisé... C'est le cas par exemple de :

- *GFS* : *Global File System* (*Red Hat*),
- *OCFS2 : Oracle Cluster File System*,
- *Lustre* (*Sun*)

- *GPFS General Parallel File System* (*IBM*)

Un système de fichier est généralement représenté sous forme hiérarchique (arborescence), il contient 3 entités :

- Le *superbloc* : décrit l'état d'occupation des secteurs alloués au système de fichiers,
- Les *inodes*,
- Les fichiers de données

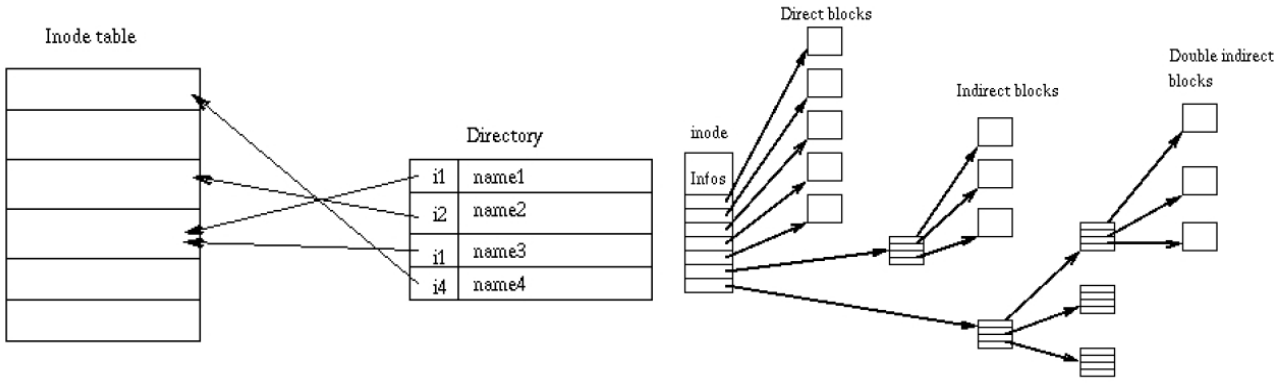


Illustration 14: Structure des inodes pour le système de fichiers ext2

Le superbloc décrit l'état d'occupation des secteurs alloués au système de fichiers, il contient :

- La taille (en blocs) de la liste des *inodes*,
- la taille (en blocs) du système de fichiers,
- le nom du système de fichiers,
- la liste des blocs et des *inodes* libres,
- le nombre de blocs et *d'inodes* libres

Les méta-donnée sont définies par des *inodes*, éléments fondamentaux des systèmes de fichiers. Ces *inodes* contiennent les informations sur les caractéristiques des fichiers

- les droits d'accès du fichier,
- propriétaire et groupe propriétaire,
- la taille du fichier,
- les dates (*MAC : modification, access, change + deletion time*),
- Le nombre d'autres *inodes* pointant vers ce fichier,
- le type de fichier (*fifo*, lien symbolique, *devices*, *pipe*...)

Il faut différencier :

- taille d'un bloc physique du disque,
- unité d'allocation (*cluster*) du système de fichier,
- la taille du fichier et l'espace occupé par le fichier sur le disque

En effet, un fichier de taille 10 octets occupera à lui seul la taille d'un bloc (512 octets, 1Ko, 2Ko...)

6 Contenu du système de fichiers Linux

6.1 Les types de fichiers

Les différents types de fichiers sous Linux sont :

- les **fichiers normaux** : ce sont des collections d'octets. Il n'existe pas de différence entre les fichiers texte et les fichiers binaires.
- les **répertoires** : ce sont des fichiers contenant les noms des fichiers et leurs numéros d'inode.
- les **liens symboliques** : permettent de présenter une image d'un fichier sous un autre nom ou à un autre endroit sans dupliquer les données.
- les **fichiers spéciaux en mode bloc** : sont les portes sur les périphériques fonctionnant par blocs de données (ex : disques).
- les **fichiers spéciaux en mode caractère** : sont les portes vers les périphériques fournissant ou consommant les données octet par octet.
- les **tubes nommés "fifo"** : permettent à deux processus sans relation de parenté de s'échanger des données comme par un tube

6.2 Les attributs des fichiers

Pour afficher les attributs principaux des fichiers, il faut utiliser l'option `'-l'` de la commande **ls** :

```
-rw-r--r-- 2 root root 6656 Apr 15 1998  chier
prw-r--r-- 1 root root 0    Apr 15 1998  fo
brw-r--r-- 1 root root 0    Apr 15 1998 bloc
crw-r--r-- 1 root root 0    Apr 15 1998 caracteres
drwxr-xr-x 1 root root 1024 Nov 12 19:42 répertoire
```

Cet affichage présente beaucoup d'informations :

- le premier caractère donne le type du fichier :
  - `'-'` pour un fichier normal
  - `'p'` pour un fifo
  - `'b'` pour un fichier spécial en mode bloc
  - `'c'` pour un fichier spécial en mode caractère
  - `'d'` pour un répertoire
- les neuf caractères suivants donnent les droits d'accès (voir plus loin)
- le champ suivant donne le nombre de liens sur le fichier
- on trouve ensuite le nom du propriétaire et du groupe du fichier
- le champ suivant donne la taille en octets du fichier
- la date de la dernière modification est indiquée selon deux formats :
  - avec l'année pour les fichiers vieux de plus de 6 mois ou de plus d'une heure dans le futur
  - avec l'heure pour les autres cas
- enfin, le nom du fichier.

La commande **stat** permet d'afficher plus d'informations sur un fichier.

```
bash$ stat file1
File: "file1"
Size: 3562 Filetype: Regular File
Mode: (0777/-rwxrwxrwx) Uid: ( 500/ sandra) Gid: ( 500/ sandra)
Device: 8,0 Inode: 2043 Links: 1
Access: Wed Nov 18 18:52:42 1997(00000.00:26:18)
Modify: Wed Nov 18 18:52:42 1997(00000.00:26:18)
```

```
Change: Wed Nov 18 18:52:59 1997(00000.00:26:01)
```

En plus des attributs déjà vus, cette commande affiche :

- le type du fichier en toutes lettres
- les droits d'accès en numérique (voir plus loin)
- la référence du périphérique physique (device)
- le numéro d'inode
- les dates de dernier accès, dernière modification et dernier changement.

6.3 Les droits d'accès

Les neuf caractères donnant les droits d'accès s'interprètent par groupe de trois :

- le premier groupe de trois caractères donne les droits pour le propriétaire
- le deuxième groupe de trois caractères donne les droits pour les utilisateurs du groupe
- le dernier groupe donne les droits pour les autres utilisateurs

Dans un groupe, la signification des caractères est donnée, dans l'ordre, par :

- 'r' pour autoriser la lecture, '-' pour l'interdire
- 'w' pour autoriser l'écriture, '-' pour l'interdire
- 'x' pour autoriser l'exécution, '-' pour l'interdire

Par exemple, un fichier avec les droits `rwxr-x--x` peut être :

- lu, écrit et exécuté par le propriétaire
- lu et exécuté par les membres du groupe
- exécuté par les autres

La signification des droits est différente en fonction du type de fichier.

Pour un fichier :

- `'r'` permet de lire le contenu du fichier
- `'w'` permet de modifier le contenu du fichier
- `'x'` permet d'exécuter un programme

Pour un répertoire :

- `'r'` permet d'afficher la liste des fichiers
- `'w'` permet de créer et de détruire des fichiers
- `'x'` permet d'accéder aux fichiers ou d'en faire le répertoire courant

Il existe deux symboles supplémentaires, `'s'` et `'t'`, pouvant prendre la place du 'x' dans la liste des droits. Ces symboles signifient :

- `'s'` : dans le cas d'un fichier exécutable, celui-ci sera exécuté avec les droits du propriétaire ou du groupe en fonction de la place du symbole. Dans le cas d'un répertoire, tous les fichiers créés dans ce répertoire appartiendront au même groupe que celui du répertoire en question.
- `'t'` (*sticky bit*) : pour les fichiers exécutables, demande de garder le code en mémoire après l'exécution. Pour les répertoires, permet de limiter la destruction des fichiers au propriétaire du répertoire, du fichier ou au super utilisateur

Ces droits d'accès peuvent s'exprimer sous forme d'un nombre en octal composé en fonction des masques de bits donnés dans le tableau ci-après :

Symbole	Valeur numérique	
r	400	Utilisateur

	40 4	groupe autre
w	400 40 4	Utilisateur groupe autre
x	200 20 2	Utilisateur groupe autre
s	4000 2000	Utilisateur groupe
t	1000	

La commande **chmod** permet de modifier les droits d'un ou plusieurs fichiers en utilisant le mode numérique ou le mode littéral.
La commande **umask** permet de fixer les droits qui seront enlevés par défaut pour la création de nouveaux fichiers.
Cette commande est généralement lancée une fois lors de l'ouverture de session, elle est intégrée à l'interpréteur de commandes et sa syntaxe varie en fonction de celui que l'on utilise. Il est conseillé de positionner le masque en numérique pour s'affranchir des différences syntaxiques.

Les commandes **chown** et **chgrp** permettent de changer, respectivement le propriétaire et le groupe d'un fichier.

6.4 Le File Hierarchy Standard

Pour assurer la compatibilité et la portabilité, les systèmes *UNIX* respectent la norme *FHS* (***File Hierarchy Standard)***. La hiérarchie de base d'un système *Unix* est la suivante :

/	la racine, elle contient les répertoires principaux
/bin	Contient les exécutables essentiels au système, employés par tous les utilisateurs.
/boot	Contient les fichiers de chargement du noyau, dont le chargeur d'amorce.
/dev	Contient les points d'entrée des périphériques.
/etc	Contient les fichiers de configuration nécessaires à l'administation du système (fichiers *passwd, group, inittab, ld.so.conf, lilo.conf, ...*).
/etc/X11	contient les fichiers spécifiques à la configuration de X (contient **XF86Config** par exemple)
/home	Contient les répertoires personnels des utilisateurs. Dans la mesure où les répertoires situés sous */home* sont destinés à accueillir les fichiers des utilisateurs du système, il est conseillé de dédier une partition spécifique au répertoire */boot* afin de limiter les dégats en cas de saturation de l'espace disque.
/lib	Contient les bibliothèques standards partagées entre les différentes application du système.
/mnt	Permet d'accueillir les points de montage des partitions temporaires (cd-rom, disquette, ...).
/proc	Regroupe un ensemble de fichiers virtuels permettant d'obtenir des informations sur le système ou les processus en cours d'exécution.

/root			Répertoire personnel de l'administrateur root. Le répertoire personnel de l'administrateur est situé à part des autres répertoires personnels car il se trouve sur la partition racine, afin de pouvoir être chargé au démarrage, avant le montage de la partition */home*.
/sbin			Contient les exécutables système essentiels (par exemple la commande **adduser**).
/tmp			contient les fichiers temporaires
/usr			Hiérarchie secondaire
	/usr/X11R6		ce répertoire est réservé au système X version 11 release 6
	/usr/X386		utilisé avant par X version 5, c'est un lien symbolique vers /usr/X11R6
	/usr/bin		contient la majorité des fichiers binaires et commandes utilisateurs
	/usr/include		contient les fichiers d'en-tête pour les programmes C et C++
	/usr/lib		contient la plupart des bibliothèques partagées du système
	/usr/local		contient les données relatives aux programmes installés sur la machine locale par le root
		/usr/local/bin	Binaires des programmes locaux
		/usr/local/include	Fichiers d'en-tête C et C++ locaux
		/usr/local/lib	Bibliothèques partagées locales
		/usr/local/sbin	binaires système locaux
		/usr/local/share	hiérarchie indépendante
		/usr/local/src	Fichiers sources locaux
	/usr/sbin		contient les fichiers bianires non essentiels au système réservés à l'administrateur système
	/usr/share		réservé aux données non dépendantes de l'architecture
	/usr/src		contient des fichiers de code source
/var			contient des données versatiles telles que les fichiers de bases de données, les fichiers journaux (logs), les fichiers du spouleur d'impression ou bien les mails en attente.

On retrouvera cette arborescence sur toutes les distributions qui respectent ce standard. Cependant, les éditeurs de distributions Linux n'utilisent pas nécessairement les mêmes conventions pour la configuration des systèmes. Il en est de même pour les outils de configuration : chaque éditeur fournit à ses utilisateurs des outils différents qui ne sont généralement pas utilisables sur les systèmes des autres distributions.

6.5 Les manuels et aides du système

Lire le manuel n'a jamais été aussi facile que sur un système Unix. Les manuels sont intégrés au système (pas besoin de lancer des recherches sur un moteur de recherche mondial sur des milliards de pages web), tout est la. La commande utile pour rechercher de l'aide sur une commande, une fonction, une bibliothèque... est : *man [-s<section>] <nom_de_commande>*

Les manuels sont divisés en plusieurs sections :

1. Commandes utilisateur
2. Appels système
3. Fonctions de bibliothèque

4. Fichiers spéciaux
5. Formats de fichier
6. Jeux
7. Divers
8. Administration système

`apropos` et `whatis` sont des commandes POSIX qui permet de lister les manuels dont la description comprend les mots passés en arguments. Chaque page de manuel comporte une courte description; `apropos` recherche et affiche cette description pour chaque page correspondant au mot-clé qu'on lui fournie. Exemple :

```
$ apropos compiler
f77 (1) - Fortran 77 compiler
gcc (1) - GNU C and C++ compiler
pc (1) - Pascal compiler
```

La commande `info` affiche une documentation multipage et hypertextuelle destiné à aider les utilisateurs d'une interface en ligne de commande et se révélant utile lorsqu'aucun environnement graphique n'est disponible. Pour se déplacer dans les pages d'aide on utilise quelques commandes simples pour parcourir l'arborescence et suivre les références croisées. Par exemple :

- *n* va à la page suivante.
- *p* va à la page précédente.
- *u* va au nœud parent.
- *l* va au dernier nœud visité.
- Pour suivre une référence croisée, il suffit de déplacer le curseur sur un lien(mot précédé du caractère *) et d'appuyer sur la touche entrée.

En général, les pages de documentation sont dans les répertoires `/usr/share/doc.`

La commande `file` permet de connaître le type de fichier donnée en paramètre. Exemple

```
$ file /bin/ls
/bin/ls: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/Linux
2.6.9, dynamically linked (uses shared libs), for GNU/Linux 2.6.9, stripped
```

7 Les commandes Linux

7.1 Principes

Utiliser Linux consiste à se connecter sur la machine et à passer les commandes nécessaires pour accomplir la tâche que l'on s'est fixée. Il existe deux environnements de travail différents sous Linux :

- le mode texte, où l'on saisit les commandes au clavier et où le résultat s'affiche sous forme de lignes de texte sur un écran
- l'environnement *Xwindow* ; il s'agit d'un environnement multi-fenêtres en réseau avec des applications commandées par une souris.

Les commandes sont des programmes écrits avec n'importe quel langage informatique pour résoudre un problème donné. Le système est évolutif et il est facile d'ajouter de nouvelles commandes.

Une commande Linux est toujours composée de la même façon :

```
nom_commande {[}options{]} liste_de_paramètres
```

Les options sont généralement composées du signe '-' et d'un caractère ou d'un mot. Elles modifient, ou précisent, le fonctionnement de la commande. La liste des paramètres permet de spécifier la cible de la commande. Les options et les paramètres sont séparés par des caractères blancs (espace ou tabulation). Tous les programmes Linux sont lancés avec trois fichiers ouverts :

- l'entrée standard (*stdin*),
- la sortie standard (*stdout*)
- la sortie des erreurs (*stderr*).

Par défaut, l'entrée standard est le clavier et les sorties sont dirigées sur l'écran. Il est possible de rediriger l'entrée ou la sortie vers un fichier ou vers une autre commande. Ceci permet d'enchaîner des traitements sur un flot de données.

signe	**fonction**
Commande < fichier	permet de rediriger l'entrée standard depuis un fichier
Commande > ficher	permet de rediriger la sortie standard vers un fichier
Commande 2> fichier	permet de rediriger la sortie des erreurs vers un fichier
2>&1	permet de rediriger la sortie des erreurs vers la sortie standard
Commande | commande	permet de rediriger la sortie standard dans l'entrée d'une autre commande
Commande >> fichier	envoie la sortie standard de la commande dans le fichier en ajoutant les lignes à la fin du fichier

7.2 L'interpréteur de commandes

Le *shell* est l'interpréteur de commandes en mode texte du système Linux lancé à chaque ouverture de session. Ce n'est pas le seul moyen de passer des commandes, il existe un environnement graphique avec des outils interactifs, mais c'est le plus pratique pour l'administrateur.
Le *shell* permet :

- l'exécution des commandes
- la redirection des entrées et des sorties
- la substitution des noms de fichiers
- la gestion des variables d'environnement
- la possibilité de réaliser des scripts

Il existe de nombreux *shells* avec des caractéristiques différentes ; celui livré en standard dans les distributions Linux est *bash* (Bourne Again SHell). Les différences se trouvent surtout dans la syntaxe du langage de scripts et dans la gestion des variables d'environnement.

7.3 Le lancement

En fonction du mode de lancement, l'interpréteur de commandes utilise différents fichiers d'initialisation. On distingue deux modes de démarrage de l'interpréteur de commandes, qui peut être un shell de démarrage (*login*) ou un shell interactif. Pour le shell de démarrage, l'ordre de lecture des fichiers de démarrage est :

- le fichier */etc/profile*
- le fichier *$HOME/.bash_profile* s'il existe, sinon le fichier *$HOME/.bash_login* sinon le fichier *$HOME/.profile*

Lorsqu'il se termine, il exécute les commandes du fichier *$HOME/.bash_logout* s'il existe. Pour un shell interactif, le fichier *$HOME/.bashrc* est exécuté.

8 Démarrage d'un système Unix

Dans ce chapitre, nous ne traiterons pas des diverses procédures d'installation purement propriétaires d'un système *Unix*, mais nous décrirons en quelques lignes le déroulement du boot jusqu'à la seconde phase d'un boot où nous prêterons une attention plus particulière au processus **init**.

Une vue d'ensemble du processus de démarrage est la suivante :

- mise sous tension;
- initialisation des modules *CPU*,
- autotest de contrôle (mémoire, test de la logique, ...)
- recherche une unité de démarrage par défaut (unité de *boot*)
- **stage 1** : chargement du *Master Boot Record* (*MBR*) : taille < 512 octets
- **stage 2** : Chargement du *bootloader* (*GRUB, LILO...*). Ce 2eme *bootloader* permet de choisir entre plusieurs systèmes d'exploitation installés sur la machine.
- charge le noyau en mémoire et lui donne la main (exécution du code préalablement chargé)
- le noyau vérifie et valide la configuration matérielle des périphériques
- activation de l'espace de pagination (*swap*)
- le noyau lance le processus ***init***.
- Le processus ***init*** traite le fichier */etc/inittab*
- le fichier */etc/inittab* exécute le fichier */etc/rc.sysinit*
- le fichier */etc/rc.sysinit* contient les appels aux instructions qui chargent les modules du noyau et initialise les drivers du matériel
- le fichier */etc/inittab* exécute le fichier */etc/rc.d/rc?.d* en fonction du ***runlevel*** choisi au boot
- le fichier */etc/inittab* exécute le fichier */etc/rc.local*
- fin du boot

8.1 Processus Init

init est le premier processus démarré et est le père de tous les futurs processus; son PID est égal à **1**. La commande *init* lit le contenu du fichier **/etc/inittab** pour démarrer complètement le système. Les processus spécifiés seront démarrés en fonction des conditions indiquées dans le fichier *inittab*. Le système sera configuré en fonction d'un niveau d'utilisation (appelé **runlevel**) qui définira quels groupes de processus seront activés (par exemple le mode multi-utilisateurs et le mode maintenance ne démarreront pas les mêmes processus).

Pour déterminer le *runlevel*, *init* va chercher dans le fichier *inittab* la présence d'une entrée du type **initdefault**. S'il ne la trouve pas, il la demandera sur la console système. Le fichier *inittab* a pour format :

`identificateur: runlevel:action:commande # Commentaire`

où :

identificateur est une chaine de caractères (14 caractères maximum) associée à l'action.

runlevel définit à quel niveau l'identificateur sera pris en compte :

0 à **6** différentes conditions possibles;
Q, q seront utilisées pour forcer *init* à relire le fichier inittab;
S, s mode mono-utilisateur (à des fins de maintenance).

action manière dont sera traité l'identificateur. Les actions les plus courantes sont :

initdefaut définit le niveau par défaut;
respawn le processus sera créé s'il n'existe pas;
wait le processus est créé une seule fois et on attend sa fin;
once le processus n'est créé qu'une seule fois;

sysinit les processus sont exécutés avant de libérer la console.
commande le script-shell ou le module à exécuter

Exemple (extrait d'une scientific linux 5):

```
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

# Standard shell at runlevel 1
~~:S:wait:/sbin/sulogin

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
co:12345:respawn:/sbin/agetty ttyS0 9600 vt100-nav
```

Ici le niveau par défaut est égal à 3. On observe que l'identificateur *co* est activé à tous les niveaux de 1 à 5. D'autre part, les commandes *mingetty* (commande d'activation de ligne de terminal) sont traités en *"respawn"* car en cas de mort du processus *mingetty,* on souhaite en voir redémarrer un nouveau pour conserver l'accès au système via la console.
Les identificateurs *rc* sont tous en "wait" car il est nécessaire que l'exécution des commandes associées soient entièrement terminées pour pouvoir passer à l'étape suivante (on ne pourra pas utiliser les services lancés en 6 avant que ceux lancés en 0 ne soient configurées).

Lorsqu'on fonctionne à un *runlevel* donné, il est à tout instant possible d'en changer sa valeur par la commande **telinit**.

telinit S

Va signaler le processus *init* et le forcer ici à passer en mode maintenance. Ceci est évidemment à proscrire en exploitation normale d'un système car *init* va brutalement tuer les processus actifs et ne plus redémarrer que ce qui correspond au runlevel *S* dans le fichiers *inittab*.

telinit 0
Va lancer un « shutodwn » du système

telinit 6
Va faire rebooter le système.

8.2 Scripts de démarrage

Sous Linux et dans la quasi totalité des systèmes UNIX récents, les services à démarrer pour un *runlevel* particulier sont placés, sous forme d'appel à des scripts, dans des dossiers */etc/rc.d/rcX.d* ou X est le niveau de démarrage. En fait on ne trouve dans ces dossiers que des liens vers les scripts de gestion des services qui sont tous placés dans le dossier */etc/init.d.*

Les liens sont nommés selon la règle suivante :
- une lettre S ou K selon qu'il s'agit de démarrer (**S**) ou de stopper (**K**) un service
- un numérique qui joue le rôle d'ordre de démarrage : *S90xxx* démarrera après *S50xxx*.
- Le nom du service

Ainsi *S10network* démarre le réseau et *S55sshd* démarre le serveur *ssh*. Le réseau démarre logiquement avant le lancement du serveur ssh.

Ces scripts sont lancés par le script */etc/rc.d/rc* avec en paramètre le « runlevel ». Les scripts **K** sont d'abord exécutés, puis les **S**.

A noter que les scripts présents dans */etc/init.d* peuvent être utilisés pendant « la vie » de la machine redémarrer un service. Ils s'emploient dans ce cas avec les paramètres **start, stop** ou **restart** : par exemple pour redémarrer ssh, on passera la commande
/etc/init.d/sshd restart

Enfin il faut noter également que la commande **chkconfig** permet de lister les démarrages ou arrêt des services, et éventuellement de les positionner. Elle crée les liens nécessaires.
Exemple :

```
# chkconfig --list snmpd
snmpd      0:off  1:off  2:off  3:off  4:off  5:off  6:off
#
# chkconfig --level 5 snmpd on
#
# chkconfig --list snmpd
snmpd      0:off  1:off  2:off  3:off  4:off  5:on  6:off
#
# ls -l /etc/rc5.d/*snmp*
lrwxrwxrwx   1  root     root          15  Jun  8  23:13  /etc/rc5.d/S50snmpd  ->
../init.d/snmpd
```

```
#
# more /etc/init.d/snmpd
#!/bin/bash
# ucd-snmp init file for snmpd
#
# chkconfig: - 50 50
# description: Simple Network Management Protocol (SNMP) Daemon
#
```

8.3 Xinetd

Toutes les applications réseaux n'ont pas un démon spécifique lancé en machine. Ainsi il existe des démons *nfsd* ou *sendmail* mais forcément pas de démons *ftpd* ou *telnetd*. Pour toutes ces dernières c'est le démon ***inetd*** (*internet daemon*) qui prend en compte l'appel; *xinetd* est le serveur réseau principal; il tourne en permanence sur une machine UNIX. Pour toutes les requêtes réseaux arrivant sur les ports qu'il est chargé de gérer, il lance un process fils qui est l'application appelée.
Par exemple lorsqu'un appel arrive sur le port 23, *xinetd* lance un process *telnetd*.
L'ensemble des applicatifs réseaux que doit gérer *xinetd* sont décrits dans son fichier de configuration /`etc/xinetd.conf` et tous les fichiers dans `/etc/xinetd.d/*.` Si un applicatif est commenté par un # en début de ligne, c'est qu'il n'est pas autorisé sur cette machine. Des applicatifs pouvant présenter des failles de sécurité, comme *tftp*, ne sont pas forcemment activés.

On dispose toujours d'un fichier de configuration général */etc/xinetd.conf* mais chaque service actif doit avoir un fichier de configuration spécifique dans */etc/xinetd.d.*
Ainsi si vous souhaitez activer un service ftp, il y aura un fichier */etc/xinetd.d/ftp* ou vous affinerez la configuartion de ce service.

```
#
# cat /etc/xinetd.conf
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
    instances        = 60
    log_type        = SYSLOG authpriv
    log_on_success     = HOST PID
    log_on_failure     = HOST
}

includedir /etc/xinetd.d

#
# cat /etc/xinetd.d/wu-ftpd
# default: on
# description: The wu-ftpd FTP server serves FTP connections. It uses \
#    normal, unencrypted usernames and passwords for authentication.
service ftp
{
    socket_type       = stream
    wait          = no
    user          = root
    server         = /usr/sbin/in.ftpd
    server_args       = -l -a
    log_on_success     += DURATION USERID
    log_on_failure     += USERID
    nice          = 10
    disable         = no
}
#
```

Si le fichier *inetd.conf* ou d'un des fichiers dans `/etc/xinetd.d/*` est modifié, on peut le faire relire au démon *xinetd* actif par la commande **`killall -HUP xinetd`**

La commande :
`# killall -HUP xinetd`
envoie un signal n° 1 à tous les processus en cours d'exécution dont le nom est *xinetd*

La liste des signaux peut être obtenue en lisant le manuel :

```
# man 7 signal
...
       Signal     Value     Action   Comment
       ----------------------------------------------------------------------
       SIGHUP        1       Term    Hangup detected on controlling terminal
                                     or death of controlling process
       SIGINT        2       Term    Interrupt from keyboard
       SIGQUIT       3       Core    Quit from keyboard
       SIGILL        4       Core    Illegal Instruction
       SIGABRT       6       Core    Abort signal from abort(3)
       SIGFPE        8       Core    Floating point exception
       SIGKILL       9       Term    Kill signal
       SIGSEGV      11       Core    Invalid memory reference
       SIGPIPE      13       Term    Broken pipe: write to pipe with no readers
       SIGALRM      14       Term    Timer signal from alarm(2)
       SIGTERM      15       Term    Termination signal
...
```

Les autres signaux souvent utilisés sont : *TERM* (n°15) et *KILL* (n°9).

On peut voir en analysant tous les process dont le *PID* père est celui de *xinetd* que par exemple pour chaque connexion *ftp*, *xinetd* crée un process *ftp*.

```
# ps -ef | grep 2902
root       2902      1   0 May14 ?          00:00:00 xinetd -stayalive -pidfile
/var/run/xinetd.pid
root     25872  2902  0 18:07 ?        00:00:00 ftpd -l -a
root     25873  2902  0 18:07 ?        00:00:00 rsync
root     25889 24347  0 18:08 pts/3    00:00:00 grep 2902
```

9 Gestion des utilisateurs et des groupes

Le système, dès son installation, avant même la première connexion au système a créé des users système.
Un utilisateur n'est donc pas uniquement une personne physique, le système a besoin d'utilisateurs pour sa gestion interne, notamment comme propriétaire des divers processus.
La commande `ps aux | less` montre qu'avant toute connexion d'utilisateur humain (repérée par les lignes `login --user`), `root` a lancé `init`, et la plupart des services, `crond, inetd, lpd, smbd, ...` , avant de lancer les connexions utilisateurs dans les consoles, y compris éventuellement la sienne !

Si l'identification des comptes et l'authentification des utilisateurs est locale, tout ce qui concerne la gestion et l'authentification des utilisateurs est inscrit dans un seul fichier `/etc/passwd`. La gestion des groupes est assurée par `/etc/group`

Les mots de passe cryptés sont maintenant placés dans `/etc/shadow`, par sécurité lisible

seulement par `root`.

Structure de */etc/passwd*. Ce fichier comprend 7 champs, séparés par le symbole :

1. nom de connexion
2. ancienne place du mot de passe crypté
3. numéro d'utilisateur **uid**, sa valeur est le véritable identifiant pour le système Linux; l'uid de `root` est 0, le système attribut conventionnellement un uid à partir de 500 aux comptes créés.
4. numéro de groupe **gid**, dans lequel se touve l'utilisateur par défaut; le `gid` de `root` est 0, des groupes d'utilisateurs au delà de 500
5. nom complet, il peut être suivi d'une liste de renseignements personnels (cf *chfn*)
6. répertoire personnel (c'est également le répertoire de connexion)
7. *shell*, interpréteur de commandes (par défaut */bin/bash*)

Structure de */etc/group.* Ce fichier comprend 4 champs, séparés par le symbole :

1. nom du groupe
2. x pour remplacer un mot de passe non attribué maintenant
3. numéro de groupe, c-à-d l'identifiant **gid**
4. la liste des membres du groupe

Les principales commandes utilisables pour la gestion des comptes :

useradd, usermod, userdel	gestion des comptes utilisateur
groupadd, groupmod, groupdel	gestion des groupes
pwck, grpck	vérification des fichiers
passwd	changer le mot de passe d'un utilisateur
chfn	Changer les informations du compte (nom complet, téléphone, bureau...)
id	Visualisation de l'identité d'un utilisateur
finger	Obtenir des informations sur le compte d'un utilisateur
chsh	Changement du shell d'un utilisateur
groups	Affiche la liste des groupes de l'utilisteur

9.1 Gestion des utilisateurs

Créer un compte pour un nouvel utilisateur : Cela signifie lui permettre d'être connu du poste local, s'y loguer, avoir un accès complet sur son répertoire. personnel.
Mais aussi dans une configuration réseau, de pouvoir se connecter à son compte par ssh et ftp, et de pouvoir bénéficier de services réseau de partage distant (sous Linux par NFS.

Pour créer l'utilisateur `stagex`, `root` passe la commande : `useradd stagex`
Ceci crée :

- le répertoire personnel `/home/stagex`, portant par défaut le nom du compte
- une nouvelle entrée dans les 2 fichiers fondamentaux `/etc/passwd` et `/etc/group`.

Pour lui attribuer le mot de passe : `passwd stagex`

Supprimer le compte d'un utilisateur (non connecté), au hasard .. `totox`.

```
userdel [-r] totox
```

L'option **-r** supprime aussi le répertoire personnel et les fichiers de l'utilisateur
La commande supprime toute trace de l'utilisateur dans le fichier de configuration : `/etc/passwd` y compris dans les groupes d'utilisateurs.

Modifier le compte de l'utilisateur toto : **`usermod [options] totox`**
`usermod -G stagiaire,prof stagex` ajoute `stagex` dans les 2 groupes stagiaire et profs (qui doivent exister)

9.2 Les groupes

Un groupe est, aussi pour Linux, un ensemble d'utilisateurs qui partagent les mêmes fichers et répertoires. Nous verrons que les fichiers accordent des droits d'accès réglables à ces groupes.

Chaque utilisateur doit faire partie au moins d'un groupe, son ***groupe primaire***. Celui-ci est défini au moment de la création du compte, et *par défaut*, l'utilisateur appartient à un nouveau groupe créé, portant son nom. Ainsi, dans `/etc/passwd` chaque utilisateur possède un groupe par défaut, précisé par son identifiant `gid` dans ce fichier.

L'appartenance au groupe primaire n'étant pas exclusive, **tout utilisateur peut faire partie de plusieurs autres groupes**, appelés ses *groupes secondaires*. Mais le rôle joué par le groupe primaire demeure prépondérant.

Pour lister tous les groupes (primaire et secondaires) d'un utilisateur :

```
groups stagex
```

Pour créer un nouveau groupe

```
groupadd stagiaire
```

Supprimer un groupe, au hasard .. encore `totox`.

```
groupdel totox
```

Le groupe est supprimé du fichier `/etc/group`.

Pour ajouter un utilisateur à un groupe : Le plus simple est d'éditer le fichier `/etc/group` et d'ajouter une liste d'utilisateurs (séparés par des virgules) sur la ligne du groupe

10 Automatisation de tâches

Linux fournit un mécanisme à l'administrateur système pour automatiser les tâches répétitives. Le programme *cron* est utilisé pour planifier ces tâches et pour fournir un compte rendu d'exécution éventuel. La commande **crontab** est utilisée pour définir les tâches que doit exécuter **cron**.

Le démon est lancé au démarrage du système et regarde, toutes les minutes, dans le fichier `/etc/crontab` et le répertoire `/var/spool/cron` s'il y a des tâches à exécuter.

10.1 Le fichier crontab

Un fichier crontab contient des instructions pour le démon cron. Chaque utilisateur possède son propre fichier crontab dans le répertoire `/var/spool/cron`. Il n'est pas éditable directement mais doit être modifié à l'aide de la commande **crontab**. Les lignes de ce fichier contiennent soit des initialisations de variables d'environnement, soit des commandes pour le démon **cron**.

Les variables `$LOGNAME`, `$HOME`, `$SHELL` sont initialisées automatiquement.

Une ligne de commande de **cron** est constituée :
- de cinq champs de définition de temps
- d'un identificateur d'utilisateur (uniquement pour les fichiers crontab du système)
- d'une commande à exécuter

Les champs de définition des temps représentent dans l'ordre :
- les minutes (valeurs de 0 à 59)
- les heures (valeurs de 00 à 24)
- les jours (valeurs de 1 à 31)
- les mois (valeurs de 1 à 12)
- le jour de la semaine (valeurs de 0 à 7, Dimanche est le 0 et le 7, ou sun, mon, tue, ?)

Chacun de ces champs peut prendre une des valeurs suivantes :
- le caractère "*" qui signifie "n'importe quand"
- une valeur numérique
- une liste de valeurs séparées par des virgules
- deux valeurs séparées par un tiret pour définir un intervalle

Le résultat de la commande doit être envoyé dans un fichier car cron ne dispose d'aucun terminal pour afficher le résultat. Si des sorties sont générées, cron les envoie par courrier à l'utilisateur.

10.2 La commande crontab

La commande **crontab** permet de modifier la liste des commandes **cron** d'un utilisateur donné. Elle peut être utilisée de deux façons :
- soit en passant un fichier en paramètre : *crontab fichier*
- soit en mode interactif : *crontab -e*

La première méthode remplace le fichier crontab de l'utilisateur par le contenu du fichier donné en paramètre. C'est la méthode utilisée dans un shell script. La deuxième méthode permet de modifier une ligne donnée du fichier crontab existant sans remettre en cause les autres lignes de commande.

10.3 La commande at

Cette commande permet de demander l'exécution d'un travail à une date donnée. Elle lit la liste des commandes à exécuter sur son entrée standard. Si les commandes génèrent des sorties, celles-ci sont envoyées sous forme d'un mail. Les spécifications de date sont très nombreuses et offrent de nombreuses possibilités. La commande **atq** permet de visualiser la liste des travaux en attente.

11 Le réseau sous Linux

11.1 Les fichiers de configuration réseau

Sur les distributions Linux du type RedHat, le fichier décrivant l'adresse IP de la machine de l'interface eth0 est */etc/sysconfig/network-scripts/ifcfg-eth0*, il contient par exemple pour une configuration manuelle (adresse IP statique) :

```
ONBOOT=yes
DEVICE=eth0
TYPE=Ethernet
BOOTPROTO=static
IPADDR=134.158.81.228
NETMASK=255.255.255.240
```

```
BROADCAST=134.158.81.239
```

Pour une configuration automatique par DHCP, le fichier */etc/sysconfig/network-scripts/ifcfg-eth0* contient :

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

Sous *Linux RedHat* les fichiers de configurations du réseau sont situés dans */etc/sysconfig*. Notamment le fichier ***network*** contient le « *hostname* » de la machine, le nom de domaine NIS….
Dans le dossier */etc/sysconfig/network-scripts* le fichier ***ifcfg-eth0*** contient entre autre le mode d'attribution de l'adresse IP et sa valeur si elle est fixée. Ainsi que le masque de sous-réseau et la passerelle.

```
# cd /etc/sysconfig/
# more network
HOSTNAME=lyoserv1.in2p3.fr
NETWORKING=yes
GATEWAY=134.158.81.225
NISDOMAIN=lyoipnl
```

Le système Unix est installé sur une machine sur le réseau. Pour pouvoir se connecter à un service de cette machine, on peut s'y connecter en mentionnant son nom ou son adresse IP. Lorsqu'une requête réseau est effectuée avec une adresse littérale, il y a un mécanisme de substitution par l'adresse numérique qui sera celle véhiculée par le protocole IP. Cette résolution se fait soit par le fichier *UNIX* **/etc/hosts** qui contient des correspondances entre adresses littérales et numériques. Soit par le mécanisme des serveurs de noms qui fera l'objet d'un chapitre spécifique. Soit éventuellement par les *NIS* (voir chapitre).

Le fichier */etc/hosts* à la forme suivante :

```
# @(#)$Header: hosts,v 1.8.194.1 93/03/26 15:57:40 kcs Exp $
#
# The form for each entry is:
# <internet address>  <official hostname> <aliases>
#
# For example:
# 192.1.2.34  hpfcrm loghost
#
# Sée the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
#    The format described in this file is the correct format.
#    The original Berkeley manual page contains an error in
#    the format description.
#
127.0.0.1    localhost    loopback
134.158.138.50 lyopsr
```

Le propre nom de la machine figure généralement dans ce fichier. On peut donner des pseudonymes à des machines (alias). L'adresse `127.0.0.1` est systématiquement réservée au bouclage réseau sur soi-même, pour l'utilisation des applications réseau vers sa propre machine (ex : *telnet* en local).

Si l'un des mécanismes ne fournit pas l'information, un autre est utilisé. Sur certains UNIX et sur Linux on fixe l'ordre du mécanisme avec lequel la résolution en adresse IP numérique sera effectuée par le fichier /**etc/nsswitch.conf** :

```
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.

# the following two lines obviate the "+" entry in /etc/passwd and /etc/ group.
passwd:   files nis
group:   files nis

#-----------------------------------------------
hosts:   dns files
networks:  files
protocols: files
rpc:    files
ethers:   files
netmasks:  files
bootparams: files
```

Il apparaît que pour la résolution des noms de "hosts" on fait d'abord appel au DNS puis, le cas échéant, au fichier consacré à cette notion (soit ici le fichier */etc/hosts)*.

Le fichier */etc/resolv.conf* : sera décrit dans le chapitre *DNS*. Citons néanmoins la ligne « *domainname* nom_de_domaine » qui permet de préciser le nom du domaine dans lequel est la machine. Et la ligne « *search* nom_de_domaine » qui permet de préciser un domaine de recherche pour lequel on pourra joindre la machine sans préciser son nom complet. Si *search* n'est pas précisé ceux sont les noms du même domaine que la machine qui peuvent être utilisés en format court.

```
# cat /etc/resolv.conf
domainname in2p3.fr
search in2p3.fr
nameserver 134.158.138.50
```

Les modifications au fichier */etc/resolv.conf* sont dynamiques, c'est-à-dire prises en compte immédiatement.
Le fichier */etc/services* : vous pouvez rajouter vos propres ports applicatifs à condition de ne pas prendre des ports "bien connus"

```
# /etc/services:
# $Id: services,v 1.42 2006/02/23 13:09:23 pknirsch Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#       http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]   [# comment]
```

```
tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
rje             5/tcp                           # Remote Job Entry
rje             5/udp                           # Remote Job Entry
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
systat          11/udp          users
daytime         13/tcp
daytime         13/udp
qotd            17/tcp          quote
qotd            17/udp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp-data        20/udp
# 21 is registered to ftp, but also used by fsp
ftp             21/tcp
ftp             21/udp          fsp fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
telnet          23/udp
```

Le fichier */etc/protocols* contient le numéro des protocoles de *TCP/IP* ; ce numéro est véhiculé dans l'entête *IP*. Ce fichier n'est en général pas à modifier.

```
# /etc/protocols:
# $Id: protocols,v 1.5 2006/10/11 15:39:11 pknirsch Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
#
# See also http://www.iana.org/assignments/protocol-numbers

ip      0       IP              # internet protocol, pseudo protocol number
hopopt  0       HOPOPT          # hop-by-hop options for ipv6
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group management protocol
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
cbt     7       CBT             # CBT, Tony Ballardie <A.Ballardie@cs.ucl.ac.uk>
egp     8       EGP             # exterior gateway protocol
igp     9       IGP             # any private interior gateway (Cisco: for IGRP)
bbn-rcc 10      BBN-RCC-MON     # BBN RCC Monitoring
nvp     11      NVP-II          # Network Voice Protocol
pup     12      PUP             # PARC universal packet protocol
argus   13      ARGUS           # ARGUS
emcon   14      EMCON           # EMCON
xnet    15      XNET            # Cross Net Debugger
chaos   16      CHAOS           # Chaos
udp     17      UDP             # user datagram protocol
```

Le fichier /etc/ethers contient des correspondances adresses ethernet/adresses *IP* pour répondre aux requêtes *RARP.*
Peuvent exister des fichiers non généralisés tel */etc/netmasks* de *Solaris* qui contient le numéro de

réseau et le *netmask*.
A noter : Ces fichiers *networks*, *protocols*, *ethers* et *netmask* ne sont pas utilisés par les distributions Linux que nous utiliserons en TP.

11.2 ARP et RARP

Le mécanisme d'**ARP** (Address Resolution Protocol) consiste à obtenir la translation entre une adresse logique IP et l'adresse physique de l'interface réseau à laquelle doit être envoyée le datagramme IP. Cet ARP se fait sous forme d'une requête générale (*broadcast*) sur le réseau, demandant 'qui' est à telle adresse IP. Toutes les machines connectées recoivent, et seule la machine concernée répond et retourne son adresse réseau (par ex : ethernet). La trame contenant le datagramme IP peut alors être émise en direction de l'adresse matérielle destinataire.

Normalement une telle requête ARP est nécessaire pour chaque datagramme IP à envoyer. Pour être plus rapide et ne pas surcharger inutilement le réseau, sous UNIX un cache d'ARP est maintenu. C'est une table de correspondance adresse IP - adresse réseau (ex : ethernet), qui se complète au fur et à mesure des requêtes ARP et où les informations restent stockées plusieurs dizaines de minutes. Lors d'une requête ARP, cette table est consultée en premier et seulement si une adresse n'y figure pas, le broadcast est émis sur le réseau. Dans ce cas l'adresse récupérée sera bien sur ajoutée à la table.

On peut accéder à la table d'ARP par la commande '**arp**' :

% arp -a : liste la table active

% arp -d : permet d'effacer des informations de la table; utile si ces infos viennent de changer (ex : changement d'une PROM ethernet d'une machine); nécessite d'être en mode privilégié.

Dans l'exemple suivant, on voit comment se complète la table d'ARP :

```
# arp -a
lyomail.in2p3.fr (134.158.138.12) at 00:08:74:A5:8D:37 [ether] on eth0
lyonas2.in2p3.fr (134.158.138.112) at 02:A0:98:03:D5:88 [ether] on eth0
lyopcs9.in2p3.fr (134.158.138.13) at 00:07:E9:04:E8:9F [ether] on eth0
Lyon-IPNL.in2p3.fr (134.158.136.1) at 00:0D:29:C6:49:40 [ether] on eth0
lyodns.in2p3.fr (134.158.138.50) at 00:11:43:DD:CD:00 [ether] on eth0

# host lyoftp.in2p3.fr
lyoftp.in2p3.fr is an alias for lyopcs1.in2p3.fr.
lyopcs1.in2p3.fr has address 134.158.138.1

# lftp lyoftp.in2p3.fr
lftp lyoftp.in2p3.fr:~> pwd
ftp://lyoftp.in2p3.fr
lftp lyoftp.in2p3.fr:~> quit

# arp -a
lyomail.in2p3.fr (134.158.138.12) at 00:08:74:A5:8D:37 [ether] on eth0
lyonas2.in2p3.fr (134.158.138.112) at 02:A0:98:03:D5:88 [ether] on eth0
lyopcs9.in2p3.fr (134.158.138.13) at 00:07:E9:04:E8:9F [ether] on eth0
Lyon-IPNL.in2p3.fr (134.158.136.1) at 00:0D:29:C6:49:40 [ether] on eth0
lyodns.in2p3.fr (134.158.138.50) at 00:11:43:DD:CD:00 [ether] on eth0
lyopcs1.in2p3.fr (134.158.138.1) at 00:06:29:1F:58:B4 [ether] on eth0
```

Le mécanisme de **RARP** (Reverse ARP) est moins utilisé et réalise l'opération inverse : à partir d'une adresse matérielle, obtenir l'adresse IP. Ce mécanisme sert essentiellement à des matériels

sans disque ou sans système d'exploitation sophistiqué, et qui ont besoin de télécharger par IP des informations (ex : imprimantes ou terminaux X). Ils se font alors servir par un calculateur leur adresse IP avant de pouvoir aller plus loin dans leur téléchargement (voir chapitre sur BOOTP).

11.3 Le routage IP

Comme nous l'avons indiqué, IP sait déterminer si la machine à atteindre est dans le même sous-réseau que l'émettrice. Sinon IP envoie le datagramme à un matériel qui joue le rôle de **routeur** (*gateway*). Ce matériel est soit une machine type station mais capable d'émettre vers 2 réseaux différents (2 interfaces), soit un matériel réseau spécialisé. Ce routeur determinera à son tour s'il doit lui-même envoyer le datagramme vers un routeur suivant et ainsi de proche en proche. C'est la technique du *saut à saut*. Un émetteur IP ne connaît jamais la totalité de la route qui sera suivie mais seulement le prochain routeur. On peut accéder ainsi à l'ensemble de l'internet mondial.

Très souvent plusieurs routes (cheminement) sont possibles pour les datagrammes et tous les fragments ne suivent pas forcément la même route. Les routeurs s'échangent entre eux des informations via des protocoles spécialisés (*RIP*, *IGRP*...) afin de maintenir des tables de routages permettant de faire 'le meilleur choix'. L'idée est de minimiser le temps (cout) total et de pouvoir palier à une défaillance.

L'intérêt est que le réseau IP est très robuste car sans nœud névralgique. L'inconvénient est que l'on ne peut pas garantir le cheminement d'une information, et donc fixer qu'elle arrive dans un temps déterminé : ceci est un problème pour tous les outils réseaux tels que vidéoconférence. L'ensemble de ces questions de routage dépassent largement ce cours et sont une affaire de spécialiste réseau. Au niveau de l'administration de machines *UNIX*, il faut savoir qu'IP a besoin de connaître une route (adresse d'un routeur) pour expédier les paquets non locaux. Deux mécanismes sont possibles pour fixer cette route : soit le *routage dynamique*, soit le *routage statique*.

Routage dynamique : Il est basé sur l'existence d'un démon qui tourne en machine et maintient une table de routage par échange d'information avec les routeurs, en utilisant les protocoles de routage. Ce démon est soit '**routed'** soit '**gated'**. *routed* est le plus standard, basé sur RIP, et est en général utilisé. *gated* est utilisé si des protocoles de routage plus spécifiques sont desirés.
routed est démarré dans les scripts de démarrage du système (différents pour chaque UNIX). Il faut savoir que *routed* utilise de façon non négligeable les ressources de la machine. Il se justifie si la configuration de routage est mouvante. Mais dans la très grande majorité des cas, on l'inhibera pour lui préférer un routage statique.

Routage statique : Il s'agit de figer l'adresse de la machine à atteindre pour sortir du sous-réseau. Cette adresse est spécifiée par la commande '**route**' placée dans un script de démarrage (ou passée en interactif). La forme la plus utilisée de *route* est :

```
route add -net default gw xxx.xxx.xxx.xxx
```

La syntaxe de la commande est :

```
route  [-v]  [-A  family] add [-net|-host] target [netmask Nm] [gw Gw]
             [metric N] [mss M] [window W] [irtt I] [reject] [mod]
             [dyn] [reinstate] [[dev] If]
```

où l'adresse IP passée en argument spécifie le routeur par défaut à atteindre. Le dernier chiffre précise un coût et est en général 1 pour une machine distante servant de route par défaut.

La commande **netstat** que nous verrons plus loin permet entre autres de voir la route définie.

La commande **traceroute**, lorsqu'elle existe, permet de connaître le chemin (de routeur en routeur) pour atteindre une destination passée en argument.

Quelques exemples de routes différentes pour partir de Villeurbanne et aller sur différentes destinations à Lyon :

```
% traceroute www.accelance.net
traceroute to www.accelance.net (213.162.49.100), 30 hops max, 38 byte packets
 1  Lyon-IPNL (134.158.136.1)  2.295 ms  0.427 ms  0.480 ms
 2  Lyon-INTER (192.70.69.13)  0.481 ms  0.464 ms  0.485 ms
 3  193.51.186.182 (193.51.186.182)  0.484 ms  0.466 ms  0.485 ms
 4  accelance-l2.peers.lyonix.net (77.95.71.10)  1.493 ms  0.967 ms  0.487 ms
 5  vlan1.titane.accelance.net (213.162.48.181)  2.488 ms  4.466 ms  1.985 ms
 6  fer.accelance.net (213.162.49.100)  3.990 ms  2.465 ms  3.996 ms

% traceroute ip-170.net-81-220-35.lyon.rev.numericable.fr
traceroute to ip-170.net-81-220-35.lyon.rev.numericable.fr (81.220.35.170), 30 hops max, 38 byte
packets
 1  Lyon-IPNL (134.158.136.1)  0.404 ms  0.307 ms  0.356 ms
 2  Lyon-INTER (192.70.69.13)  0.478 ms  0.342 ms  0.237 ms
 3  193.51.186.182 (193.51.186.182)  0.232 ms  0.218 ms  0.234 ms
 4  nri-b-pos9-0.cssi.renater.fr (193.51.179.13)  5.606 ms  5.710 ms  5.607 ms
 5  NEO-TELECOMS-ABOVENET.sfinx.tm.fr (194.68.129.201)  7.231 ms  8.220 ms  7.586 ms
 6  te1-4.er2b.cdg2.fr.above.net (64.125.23.33)  6.475 ms *  6.557 ms
 7  83.167.34.140.cbv2rj.numericable.net (83.167.34.140)  6.581 ms  20.197 ms  6.479 ms
 8  ip-93.net-80-236-6.asnieres.rev.numericable.fr (80.236.6.93)  13.724 ms  13.701 ms  13.726 ms
 9  * * *

% traceroute alyon-159-1-69-199.w90-57.abo.wanadoo.fr
traceroute to alyon-159-1-69-199.w90-57.abo.wanadoo.fr (90.57.148.199), 30 hops max, 38 byte packets
 1  Lyon-IPNL (134.158.136.1)  0.414 ms  0.431 ms  0.357 ms
 2  Lyon-INTER (192.70.69.13)  34.732 ms  0.572 ms  112.167 ms
 3  193.51.186.182 (193.51.186.182)  0.224 ms  0.214 ms  0.232 ms
 4  te-4-2.car1.Paris1.Level3.net (212.73.207.173)  6.730 ms  6.713 ms  6.729 ms
 5  ae-12-53.car2.Paris1.Level3.net (4.68.109.80)  6.859 ms  6.959 ms  6.855 ms
 6  * * *
```

11.4 ICMP – la commande PING

ICMP (*Internet Control Message Protocol*) est un protocole que l'on situe au même niveau que *IP* dans le modèle en couches. *ICMP* est chargé de véhiculer les erreurs et informations concernant *IP* (par exemple qu'un datagramme n'est pas transmis dans un temps donné), ou envoyer une requête de présence à une autre machine. Les informations *ICMP* sont empaquetées dans des datagrammes *IP*.

Nous ne traiterons pas plus profondément *ICMP* mais nous le citons car il est à la base d'une commande *UNIX* très utilisée qui est '***ping***' (pour *Packet INternet Groper*). *Ping* génère la requête *ICMP* 'echo' à destination d'une machine et indique en combien de temps (ms) celle-ci est capable de répondre. L'intérêt premier est de savoir si une machine est accessible. Le second intérêt est, en fonction du temps de réponse, de connaître son éloignement, ou bien, si elle est 'proche', de savoir qu'il y a un problème manifeste de réseau .

La commande *ping* ne se comporte pas de façon identique sur tous les *UNIX* : elle génère soit une série de requêtes successives numerotées (interruptible par *CTRL-C*) ou répond simplement '*machine is alive*'

Exemple :

```
%
% ping lyohp0
PING lyohp0.in2p3.fr (134.158.137.11): 56 data bytes
64 bytes from 134.158.137.11: icmp_seq=0 ttl=255 time=2.6 ms
64 bytes from 134.158.137.11: icmp_seq=1 ttl=255 time=0.9 ms
```

```
64 bytes from 134.158.137.11: icmp_seq=2 ttl=255 time=0.8 ms
64 bytes from 134.158.137.11: icmp_seq=3 ttl=255 time=0.8 ms
64 bytes from 134.158.137.11: icmp_seq=4 ttl=255 time=0.8 ms
64 bytes from 134.158.137.11: icmp_seq=5 ttl=255 time=0.8 ms

--- lyohp0.in2p3.fr ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.8/1.1/2.6 ms
%
% ping prep.ai.mit.edu
PING prep.ai.mit.edu (18.159.0.42): 56 data bytes
64 bytes from 18.159.0.42: icmp_seq=0 ttl=242 time=167.3 ms
64 bytes from 18.159.0.42: icmp_seq=1 ttl=242 time=240.4 ms
64 bytes from 18.159.0.42: icmp_seq=2 ttl=242 time=190.3 ms
64 bytes from 18.159.0.42: icmp_seq=3 ttl=242 time=263.6 ms
64 bytes from 18.159.0.42: icmp_seq=4 ttl=242 time=317.8 ms

--- prep.ai.mit.edu ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 167.3/235.8/317.8 ms
%
%ping lyosu3
PING lyosu3.in2p3.fr (134.158.137.56): 56 data bytes

--- lyosu3.in2p3.fr ping statistics ---
159 packets transmitted, 0 packets received, 100% packet loss
%
```

11.5 Les commandes ifconfig et netstat

L'ensemble des notions que nous venons de voir, adresse IP, subnetmask, broadcast, route, etc... est bien entendu destiné à configurer sous UNIX le bon fonctionnement de TCP/IP. Nous allons voir 2 commandes qui s'adressent plus spécifiquement à IP. La commande **ifconfig** permet à la fois de configurer l'interface réseau vis-à-vis de IP et d'afficher la configuration existante. La commande **netstat** permet de remonter des informations sur la configuration et le fonctionnement de IP. La commande **ifconfig** figurera en général dans les scripts de démarrage du système.

```
% ifconfig nom_interface_reseau [ paramètres ]
```

ifconfig sans autre paramètre que le nom d'interface renvoie la configuration actuelle.

Certaines versions permettent l'option -a, sans nom d'interface, et renvoie la liste des interfaces réseau présentes.

ifconfig permet de figer des paramètres tels que adresse IP, adresse de broadcast, adresse de masque de sous-réseau.

Exemple :

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:1D:09:0B:CC:21
          inet addr:134.158.138.202  Bcast:134.158.143.255  Mask:255.255.248.0
          inet6 addr: fe80::21d:9ff:fe0b:cc21/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:139737657 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135968414 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2853231051 (2.6 GiB)  TX bytes:2703589299 (2.5 GiB)
          Interrupt:177

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

```
RX packets:756726 errors:0 dropped:0 overruns:0 frame:0
TX packets:756726 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:40910220 (39.0 MiB)  TX bytes:40910220 (39.0 MiB)
```

La commande **netstat** admet un grand nombre d'options; il n'est pas envisageable de les détailler toutes (faire "*man netstat*")

netstat sans option, entre en mode interactif et renvoie tous les appels actifs ; la sortie sera examinée au paragraphe suivant.

netstat -i montre l'état des interfaces réseaux

netstat -r montre la route définie; les flags UGH signifient Up Gateway Host selon le rôle de la machine.

netstat -s fournit un relevé détaillé de statistiques sur les différents protocoles.

Exemple :

```
 # netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.187.0   *               255.255.255.0   U         0 0          0 vmnet8
172.16.185.0    *               255.255.255.0   U         0 0          0 vmnet1
134.158.136.0   *               255.255.248.0   U         0 0          0 eth0
169.254.0.0     *               255.255.0.0     U         0 0          0 eth0
default         Lyon-IPNL.in2p3 0.0.0.0         UG        0 0          0 eth0
root@lyopc352 ~ # netstat -in
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500   0 139766213      0      0      0 135996143      0      0      0 BMRU
lo        16436   0   756726      0      0      0   756726      0      0      0 LRU
vmnet1     1500   0        0      0      0      0       80      0      0      0 BMRU
vmnet8     1500   0        0      0      0      0       84      0      0      0 BMRU
```

11.6 TCP et UDP – Les ports

Nous avons signalé que *TCP* et *UDP* sont les 2 principaux protocoles de la couche transport, *situés* au-dessus de IP. Ils diffèrent par le fait que *TCP* est dit "orienté connexion". Cela signifie que TCP établit un contact avec la machine destinataire avant d'émettre. *TCP* découpe les données en segments et les numérote. Le destinataire connaît la taille maximum des segments et les remet dans l'ordre d'après leur rang. La machine destinataire transmettra un acquittement à chaque segment de données reçu.

UDP lui est dit non connecté. Il est, comme *IP*, orienté datagramme qu'il émet sans se soucier du résultat. Il rajoute néanmoins, comme *TCP*, une somme de contrôle sur les données, alors *qu'IP* ne l'effectue que sur l'entête. Cette somme de contrôle est obligatoire sur *TCP* et optionnelle sur *UDP*.

Sans rentrer plus dans les détails on voit que *TCP* sera plus fiable que *UDP*. Par contre *UDP* est plus rapide. Se serviront de TCP, les applications où la fiabilité est requise tel *ftp*; ou bien lorsque les contraintes de temps ne sont pas critiques tel *telnet*. Par contre, lorsque le temps de réponse est fondamental, tel l'accès à un système de fichiers distant avec *NFS*, c'est *UDP*
qui est utilisé.

Ces 2 protocoles de transport ont pour rôle : du côté émetteur de prendre en compte les données applicatives pour les transmettre à IP; du côté récepteur de récupérer les données venant de IP pour appeler et transmettre à l'application adéquate. Pour que cette application réseau soit identifiée de

façon simple, elle est estampillée par un numéro spécifique appellé **numéro de port**. *UDP* et *TCP* incluent dans leurs entêtes le numéro de port de l'application appelante et destinataire.
Pour chaque applications réseau, des tâches dites serveurs (*daemon*) "sommeillent" en machine. La tâche adéquate est "réveillée" en fonction du numéro de port *UDP* ou *TCP* appellé.

La liste de ces identifiants d'applications s'accroît bien entendu au cours du temps. La liste des ports déjà utilisés fait l'objet d'une *RFC* réactualisée régulièrement. Elle est dit liste des "ports bien connus". Par exemple *telnet* travaille avec le port 23, *ftp* avec 21 ...

Le fichier **/etc/services** contient la liste des applications réseaux avec pour chacune le couple "port/protocole de transport" qu'elle utilise. Il est détaillé au paragraphe suivant.

La commande netstat sans option montre les requêtes en cours et notamment le protocole transport et le port utilisé

```
#  netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 lyopc352.in2p3.fr:36560     lyogrid00.in2p3.fr:ssh      ESTABLISHED
tcp        0      0 lyopc352.in2p3.fr:50542     lyopcs9.in2p3.fr:ssh        ESTABLISHED
tcp        0      0 lyopc352.in2p3.fr:33345     ug-in-f91.google.com:https  ESTABLISHED
tcp        1      0 lyopc352.in2p3.fr:34317     lyopcs19.in2p3.fr:ipp       CLOSE_WAIT
tcp        0      0 lyopc352.in2p3.fr:48113     lyopcs17.in2p3.fr:http      TIME_WAIT
tcp        1      1 lyopc352.in2p3.fr:58608     lyopcs19.in2p3.fr:ipp       LAST_ACK
tcp        0      0 lyopc352.in2p3.fr:37288     lyodmz1.in2p3.fr:ssh        ESTABLISHED
tcp        1      0 lyopc352.in2p3.fr:45185     fg-in-f91.google.com:http   CLOSE_WAIT
tcp        0      0 lyopc352.in2p3.fr:52175     lyomail.in2p3.fr:imaps      ESTABLISHED
tcp       54      0 lyopc352.in2p3.fr:52173     lyomail.in2p3.fr:imaps      CLOSE_WAIT
tcp        1      0 localhost.localdomain:56902 localhost.localdom:webcache CLOSE_WAIT
tcp        0      1 lyopc352.in2p3.fr:46716     eiecb212.webex.com:https    FIN_WAIT1
tcp        0      0 lyopc352.in2p3.fr:57399     lyogrid02.in2p3.fr:ssh      ESTABLISHED
......
#
```

Avec la commande netstat, il est également possible de visualiser l'ensemble des services réseau actifs et les processus correspondants avec les bonnes options :

```
root # netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address  State    PID/Program name
tcp        0      0 127.0.0.1:2208              0.0.0.0:*        LISTEN   10520/hpiod
tcp        0      0 0.0.0.0:53857               0.0.0.0:*        LISTEN   -
tcp        0      0 0.0.0.0:111                 0.0.0.0:*        LISTEN   2842/portmap
tcp        0      0 127.0.0.1:631               0.0.0.0:*        LISTEN   10568/cupsd
tcp        0      0 0.0.0.0:952                 0.0.0.0:*        LISTEN   2882/rpc.statd
tcp        0      0 127.0.0.1:6010              0.0.0.0:*        LISTEN   6865/11
tcp        0      0 0.0.0.0:830                 0.0.0.0:*        LISTEN   10401/ypbind
tcp        0      0 127.0.0.1:2207              0.0.0.0:*        LISTEN   10525/python
tcp        0      0 :::8080                     :::*             LISTEN   21883/java
tcp        0      0 :::22                       :::*             LISTEN   10554/sshd
tcp        0      0 ::1:6010                    :::*             LISTEN   6865/11
tcp        0      0 :::5308                     :::*             LISTEN   12590/cfservd
udp        0      0 0.0.0.0:57514               0.0.0.0:*                 12632/avahi-daemon:
udp        0      0 0.0.0.0:938                 0.0.0.0:*                 2882/rpc.statd
udp        0      0 0.0.0.0:946                 0.0.0.0:*                 2882/rpc.statd
udp        0      0 0.0.0.0:827                 0.0.0.0:*                 10401/ypbind
udp        0      0 0.0.0.0:68                  0.0.0.0:*                 2440/dhclient
udp        0      0 0.0.0.0:966                 0.0.0.0:*                 10401/ypbind
udp        0      0 0.0.0.0:7001                0.0.0.0:*                 -
udp        0      0 0.0.0.0:5353                0.0.0.0:*                 12632/avahi-daemon:
udp        0      0 0.0.0.0:111                 0.0.0.0:*                 2842/portmap
udp        0      0 0.0.0.0:38513               0.0.0.0:*                 -
udp        0      0 0.0.0.0:631                 0.0.0.0:*                 10568/cupsd
udp        0      0 134.158.138.202:123         0.0.0.0:*                 10608/ntpd
udp        0      0 127.0.0.1:123               0.0.0.0:*                 10608/ntpd
udp        0      0 0.0.0.0:123                 0.0.0.0:*                 10608/ntpd
udp        0      0 :::46523                    :::*                      12632/avahi-daemon:
udp        0      0 :::5353                     :::*                      12632/avahi-daemon:
udp        0      0 fe80::21d:9ff:fe0b:cc21:123 :::*                      10608/ntpd
udp        0      0 ::1:123                     :::*                      10608/ntpd
udp        0      0 :::123                      :::*                      10608/ntpd
```

La commande ***tcpdump*** permet de capturer des paquets sur le réseau et donc par exemple d'analyser l'établissement d'une connexion tcp en cas de problème. *tcpdump* permet d'utiliser de multiples critères de captures, qu'il n'est pas pour objet de développer ici. Le manuel de la commande est très complet. Les critères les plus courant permettent de ne sélectionner qu'une machine source et/ou destination, afin d'observer les connexions souhaitées sans être pollué par la totalité des paquets. De même on peut filtrer sur un protocole particulier.

```
#
# /usr/sbin/tcpdump src host lyopcs1 and port ftp
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet socket
tcpdump: listening on all devices
10:54:10.876190 eth0 < lyopcs1.in2p3.fr.2676 > lyoforse.in2p3.fr.ftp: S 1254501825:1254501825(0) win
32120 <mss 1460,sackOK,timestamp 1441994306 0,nop,wscale 0> (DF)
10:54:10.876190 eth0 < lyopcs1.in2p3.fr.2676 > lyoforse.in2p3.fr.ftp: . 1254501826:1254501826(0) ack
1065876468 win 32120 <nop,nop,timestamp 1441994306 228403285> (DF)
10:54:11.106190 eth0 < lyopcs1.in2p3.fr.2676 > lyoforse.in2p3.fr.ftp: . 0:0(0) ack 64 win 32120
<nop,nop,timestamp 1441994329 228403308> (DF) [tos 0x10]
```

12 SSH

SSH est à la fois la définition d'un protocole et un ensemble de programmes utilisant ce protocole, destinés à permettre aux utilisateurs d'ouvrir, depuis une machine cliente, des sessions interactives à distance sur des serveurs et de transférer des fichiers entre les deux.

Parmi les avantages de *SSH* sur des applications analogues (*telnet, rlogin, ftp*):

- la connexion entre le client et le serveur est encryptée. Les pirates, même s'ils contrôlent cette connexion, ne peuvent pas lire les données confidentielles (telles que des mots de passe) qui y transitent.
- la cryptographie à clefs publiques permet à *SSH* de garantir l'authentification mutuelle du client (utilisateur) et du serveur. Des pirates ne peuvent pas dévier une connexion vers un serveur sous leur contrôle.

12.1 Chiffrement

Lorsque l'on installe le logiciel *SSH* sur un serveur, on génère un couple de clés asymétriques, l'une est dite privée, l'autre publique.

Lorsqu'un client se connecte par *SSH*, le serveur lui envoie sa clef publique, il y a alors négociation d'une clef de session, celle-ci sera chiffrée par la clef publique du serveur pour son transport sur le réseau, seul le serveur possesseur de la clef privée pourra déchiffrer cette clef de session.
Ensuite, toutes les données échangées entre les deux machines seront chiffrées et déchiffrées par l'algorithme avec la clef de session

12.2 Tunnel

Rappel : une application utilisant le réseau se connecte à un serveur donné sur un port donné, correspondant à un service spécifique. Par exemple, votre outil de courrier électronique se connecte au serveur lyoserv.in2p3.fr sur le port 25 (port *SMTP*) pour lui confier le message que vous désirez envoyer à votre correspondant.

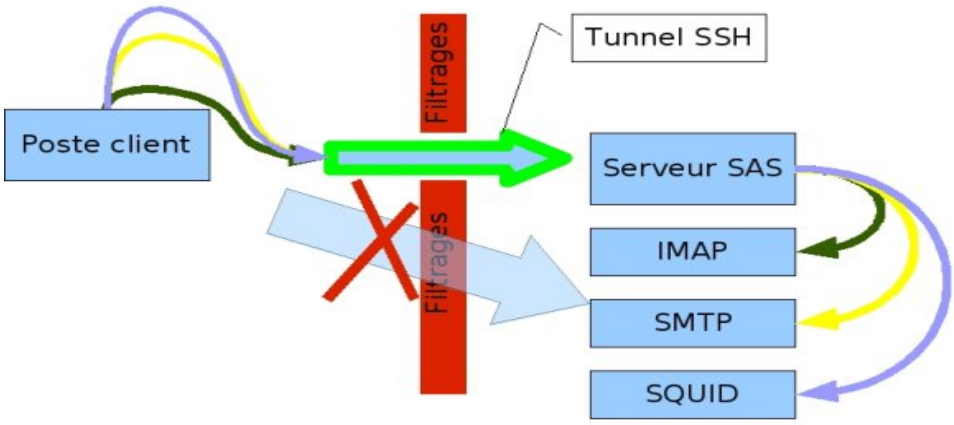


Illustration 15: Tunnels SSH

Les tunnels créés dans la figure ci-dessus pourraient l'être avec la commande ssh suivante :

```
ssh     -L10443:imap.domaine.fr:443 \
        -L10025:smtp.domaine.fr:25 \
        -L13128:squid.domaine.fr:3128 \
        user@sas.domaine.fr
```

Lne fois *user* authentifié et connecté sur la machine *sas*, il peut alors se connecter sur :

- *localhost:10443* pour que sa connexion soit relayée à travers le tunnel à la machine *imap.domaine.fr* située derrière les filtres
- *localhost:10025* pour que sa connexion soit relayée à travers le tunnel à la machine smtp.domaine.fr située derrière les filtres
- *localhost:13128* pour que sa connexion soit relayée à travers le tunnel à la machine *squid.domaine.fr* située derrière les filtres

Les clients SSH offrent la possibilité de créer des tunnels *SSH*. A travers un menu ou par des options sur la ligne de commande (selon le client utilisé), on peut spécifier le serveur et le port (service) de ce serveur qui seront la cible du tunnel. Notons que ce serveur est en général différent du serveur SSH sur lequel le client a ouvert une session interactive. Le logiciel client *SSH* va alors ouvrir ce même port sur la machine cliente *SSH* (il est possible de faire ouvrir un port différent, mais ceci est rarement utilisé dans la pratique). Quand une application se connecte à ce port, ouvert sur le client *SSH*, le logiciel *SSH* sur le client *SSH* et sur le serveur *SSH* collaborent pour que les données transmises sur ce port parviennent au serveur cible du tunnel, et inversement, les données émises par ce serveur parviennent à l'application connectée sur le port source du tunnel.

Dans le cas du protocole *X11*, c'est l'inverse, le port est ouvert sur le serveur *SSH* (qui est le client *X11*) et les données sont transmises par le tunnel au client *SSH* (qui est le serveur *X11*), la machine qui affiche est donc *localhost*.

12.3 L'authentification par clé

Face à la faiblesse de l'authentification par mot de passe, l'authentification par clé se révèle être très efficace. La clé permet de garantir à un système qu'un utilisateur est bien celui qu'il prétend être ... en deux mots :"Je jure et je prouve que c'est bien moi".L'authentification par clé fonctionne grâce à 3 composants :
- Une clé publique : elle sera exportée sur chaque hôte sur lequel on souhaite pouvoir se connecter.
- Une clé privée : elle permet de prouver son identité aux serveurs.

- Une passphrase : Permet de sécuriser la clé privée (notons la subtilité, passphrase et pas password ... donc "phrase de passe" et non pas "mot de passe").

La sécurité est vraiment accrue car la passphrase seule ne sert à rien sans la clé privée, et vice-versa.

12.4 Création de la paire de clé

La création de la paire de clé se fait avec ssh-keygen.
Il existe 2 types de clés : RSA et DSA. Chacune pouvant être de longueur différentes : 1024, 2048, 4096 bits (les clés inférieures à 1024 bits sont à proscrire ...surtout les RSA).
Pour créer une clé DSA de 1024 bits : *ssh-keygen -t dsa -b 1024 -C username@domain.tld*

Le commentaire permet de distinguer les clés, utile quand on a plusieurs clé (notamment une perso et une pour le boulot). Ici la distinction se fait sur l'adresse e-mail. Si le commentaire est omis, il sera de la forme user@host.

```
$ ssh-keygen -t dsa -b 1024 -C username@domain.tld
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
cb:61:48:6b:b4:53:00:9b:d1:2a:cf:44:88:79:c2:19 username@domain.tld
```

2 fichiers ont été crées (dans le dossier ~/.ssh/) :
- *id_dsa* (ou *id_rsa* dans le cas d'une clé RSA) : contient la clé privée et ne doit pas être dévoilé ou mis à disposition
- *id_dsa.pub* (ou *id_rsa.pub* dans le cas d'une clé RSA) : contient la clé publique, c'est elle qui sera mise sur les serveurs dont l'accès est voulu.

12.5 Paramétrage de SSHd pour autoriser les authentifications par clé

Le fichier de configuration est /*etc/ssh/sshd_config*.
Par défaut ce comportement de sshd est interdit. La ligne dans le fichier gérant l'authentification par clé est :

```
#PubkeyAuthentication yes
```

Le signe "#" signifie que la ligne est commentée. Il convient donc de la décommenter.

```
PubkeyAuthentication yes
```

Il suffit de demander au service *sshd* de relire sa configuration afin que le serveur soit en mesure d'accepter les authentifications par clé.

```
[root@serveur ~]# service sshd reload
Rechargement de sshd :                                     [  OK  ]
```

Cela ne suffit bien entendu pas car aucune clé publique n'a été envoyée sur le serveur (serveur)
L'authentification par mot de passe va être utilisée

12.5.1 Première méthode de copie d'une clé :

```
$ cat ~/.ssh/id_dsa.pub | ssh serveur "cat - >> ~/.ssh/authorized_keys"
```

```
user@serveur's password:
```

Cette commande va lire le fichier $HOME/.ssh/id_dsa.pub (clé publique), se connecter sur le serveur "serveur" et ajouter au fichier des clés autorisées ($HOME/.ssh/authorized_keys) le contenu de la clé lue.

12.5.2 Deuxième méthode de copie d'une clé :

Il s'agit du même principe que la première méthode sauf que tout est décomposé (utile pour bien comprendre) :

```
$ scp ~/.ssh/id_dsa.pub serveur:/tmp
user@serveur's password:
id_dsa.pub                                          100%  609     0.6KB/s
00:00
$ ssh serveur
[user@serveur ~]$ cat /tmp/id_dsa.pub >> /root/.ssh/authorized_keys
[user@serveur ~]$ rm /tmp/id_dsa.pub
```

12.5.3 Troisième méthode :

Il s'agit de la méthode la plus automatisée et la plus simple (elle est a utiliser quand on a compris ce qu'elle va faire ...)

```
$ ssh-copy-id -i ~/.ssh/id_dsa serveur
root@durandal's password:
Now try logging into the machine, with "ssh 'serveur'", and check in:

 .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

ssh-copy-id va copier la clé publique (pas besoin de préciser le .pub) sur l'hôte distant. Pas besoin de taper de nombreuses lignes.

- Piège a éviter :

Il faut s'assurer dans le cas ou un fichier *authorized_keys* est présent, qu'il se termine bien par une nouvelle ligne. Sinon le rajout se fera à la volée et 2 clés (la dernière et celle rajoutée) seront inutilisables.

12.6 Fichiers de configuration

Fichiers système pour le client : **/etc/ssh/sshd_config**

Coté client, le client ssh va d'abord lire le fichier de configuration **/etc/sshd/ssh_config**, puis va lire le fichier ~**/.ssh/config** qui se trouve dans le *$HOME* de l'utilisateur.
Ce fichier est de la forme :

```
% cat ~/.ssh/config
host svn.sourceforge.net
  IdentityFile .ssh/sf.net
  PasswordAuthentication no
  Protocol 2
```

```
  PubkeyAuthentication yes
  User user

host *
  ForwardX11 yes
  ForwardX11trusted yes
```

Avec ce fichier de configuration, quand on se connecte par ssh sur le serveur svn.sourceforge.net, le protocole ssh sera 2, l'authentification se fera par une clé publique (*RSA* ou *DSA*) dont la clé privée est stockée dans le fichier .ssh/sf.net, et l'utilisateur distant est user. Pour tous les autres serveurs ssh , lors de la connexion, l'activation du tunneling X11 sera demandée.

12.7 ssh-agent

Le serveur ssh est maintenant plus sécurisé, mais taper des passphrases à longueur de journée peut se révéler être très pénible surtout si on a choisi une "vraie" passphrase. L'agent ssh permet de taper la passphrase une seule fois et de la conserver en mémoire pendant tout son fonctionnement. Les communications ssh fonctionneront donc de façon transparente.

- Mode texte

Il faut lancer l'agent avec un shell (le plus simple étant de le lancer avec la variable $SHELL qui contient le shell courant)
Ensuite le programme ssh-add permet de charger les clé présentes dans ~/.ssh/. La passphrase est demandée, toutes les connexions nécessitant les clés chargées par l'agent, seront transparentes.

```
$ ssh-agent $SHELL
$ ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
$ ssh serveur
Last login: Mon Jul  3 17:21:20 2006 from serveur.home.lan
```

- Mode graphique

Prérequis : Le package openssh-askpass doit être installé (openssh-askpass-gnome dans les versions de Fedora < 5) Cela peu être vérifié en tapant rpm -q openssh-askpass.

```
$ rpm -q openssh-askpass
openssh-askpass-4.3p2-4
```

Il faut aller dans le gestionnaire de sessions : Bureau, Préférences, Préférences Supplémentaires, Sessions Une fois le gestionnaire de session lancé, il faut aller dans l'onglet "Programme au démarrage", faire "ajouter" et taper /usr/bin/ssh-add

A la prochaine ouverture de session, une fenêtre demandera les passphrases de chaque clé trouvée dans ~/.ssh/

12.8 Troubleshooting

12.8.1 Première connexion

Lors de la première connexion ssh vers un hôte, ce message peut apparaître :

```
The authenticity of host 'serveur (192.168.24.3)' can't be established.
RSA key fingerprint is c6:ee:c6:e4:9a:b6:7e:46:4c:17:b4:d0:7b:80:af:2c.
Are you sure you want to continue connecting (yes/no)?
```

Il faut répondre oui lors de cette première connexion.

```
Warning: Permanently added 'serveur' (RSA) to the list of known hosts.
```

La clé d'hôte sur le serveur est maintenant conservée (fichier ~/.ssh/known_hosts) .

12.8.2 Changement de clé d'hôte sur le serveur

En cas de changement de clé le message suivant apparaitra :

```
# ssh serveur
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
c6:ee:c6:e4:9a:b6:7e:46:4c:17:b4:d0:7b:80:af:2c.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending key in /home/user/.ssh/known_hosts:12
RSA host key for serveur has changed and you have requested strict checking.
Host key verification failed.
```

Un changement de clé peut avoir plusieurs cause :
- La clé de l'hôte a été changée volontairement par un administrateur
- Le serveur a subi une réinstallation sans sauvegarde de ses clés
- Ce n'est plus le même serveur (et il faut donc se renseigner si c'est normal ... car cela peut être le serveur d'un pirate ...)

Pour corriger le problème (en cas de changement non suspect uniquement !!!), il suffit soit :
- De supprimer la clé du serveur dans ~/.ssh/known_hosts (ici dans la 12eme place)
- De la modifier dans ~/.ssh/known_hosts afin de mettre la nouvelle clé de l'hôte si elle est connue

12.8.3 Les clés ne fonctionnent pas

Il faut vérifier les droits sur les clés, ssh nécessite que le dossier .ssh ait les permissions 700 (rwx pour l'utilisateur, rien pour le groupe et les autres) , les clés privée ainsi que authorized_keys 600 (r pour l'utilisateur, rien pour le groupe et les autres) et les clés publiques 644 (rw pour l'utilisateur, read pour le groupe et les autres).
Il faut aussi vérifier que chaque clé tiens sur une ligne dans authorized_keys (même si elle apparait sur plusieurs ligne a l'écran, dans le fichier elle correspond qu'à une ligne).

13 SYSLOG

C’est un démon qui permet de stocker des messages afférents au fonctionnement du système.
Les messages sont classés par **niveau** et **fonctionnalité**.

En partant de la sévérité maximale, les différents niveaux sont les suivants:

- **EMERG**. Situation de panique
- **ALERT**. Intervention requise (base système corrompue)
- **CRIT**. Par exemple: erreur matérielle sur disque

- **ERR**. Des erreurs
- **WARNING**. Messages d'attention
- **NOTICE**.
- **INFO**. Comme NOTICE, messages de moindre importance
- **DEBUG**. Info de débogage de certains programmes

Parmi les fonctionnalités, on trouve:

- **KERN**. Qui s'interesse aux messages générés par le noyau
- **USER**. Pour les messages générés par des processus utilisateurs
- **MAIL**. Pour les messages liés au système de mail
- **DAEMON**. Pour les démons système: routed, ftpd, rshd,...
- **AUTH**. Pour le système d'authentification: login, su, getty mais également par ftp ou rsh
- LPR. Par le système d'impression :lpr, lpd, etc...
- **LOCAL0 à LOCAL7**. lesquelles peuvent être gérées depuis des applications "maison".

Les messages générés par les applications qui se conforment à la mode syslog, peuvent être stockés dans un fichier, affichés sur un terminal ou redirigés vers une autre machine. Le démon **syslogd** est en charge de l'acheminement des messages à bon port en fonction des conditions précisées dans le fichier de configuration **/etc/syslog .conf.**

13.1 Le fichier /etc/syslog.conf

Ce fichier définit les règles appliquées pour le stockage des messages. Ces règles sont constituées par un couple (fonctionnalité.sevérité) associé à la destination du message. Attention, ces champs doivent impérativement être séparés par une ou plusieurs tabulation (l'utilisation d'un caractère d'espace est interdite). La règle précise que pour une fonctionnalité donnée, tous les messages d'une sévérité supérieure à celle mentionnée sont consignés Par exemple :

```
auth.debug        /prot/logs
```

consignera les messages de type « auth » d'une sévérité supérieure ou égale à « débug » dans le fichier */prot/logs*.

Des caractères génériques peuvent être utilisés pour simplifier l'écriture. Voici quelques exemples

```
*.alert          /var/log/alertes
```

toutes les fonctionnalités à partir du niveau alerte sont consignées dans le fichier /var/log/alertes.

```
mail.=info     /var/adm/syslog
```

indique que seule la sévérité "info" est concernée.

```
kern.
emerg          *
```

indique que tous les utilisateurs connectés sont informés.

13.2 Syslog en réseau

Il peut être intéressant de centraliser touts les messages des machines d'un réseau local en seul point. Chaque machine dispose d'un fichier de configuration minimum qui indique à quelle machine envoyer les messages, cette dernière prenant en charge l'aiguillage des messages selon les conditions souhaitées.

Ceci pourrait être le contenu du fichier syslog .conf sur chaque machine du réseau :

```
*.info,mail.none    /var/adm/syslog
*.notice        @addr_serveur
```

tous les messages d'une sévérité supérieure à *notice* étant envoyés par réseau à la machine dont l'adresse est *addr-serveur*, cette dernière effectuant alors le tri.

Sur les systèmes Linux, il convient d'activer cette possibilité en ajoutant l'option **« -r »** dans les options par défaut du demon *syslogd*.
Pour ce faire, modifiez le fichier **/etc/sysconfig/syslog**.

13.3 Le journal des evènements

Le fichier journal des evènements *syslogd* ressemble à l'exemple ci-dessous :

```
Feb 19 01:06:29 ccpnxa12 rcp[39144]: EXPORT file to host cctms cmd  is rcp -t
/tmp/cf.cctms, local user root.
May 14 11:08:30 ccpnds00 last message repeated 2 times
Feb 19 01:19:44 ccpnds00 syslog: NOTICE, load average is OK.
Feb 20 17:24:42 ccpnds00 syslog: User oneall used rootacc
Feb 20 18:00:58 ccpntc15 login[39853]: User delaunay logged in on port
  /dev/pts/ from host ccapollo.in2p3.fr.
Feb 20 18:01:41 ccpnds00 rlogind[54323]: connect from h2.lyon.fr
Feb 20 18:04:03 ccali25 su: BAD SU from to root at /dev/pts/1
Feb 20 18:04:35 ccpnds00 login[43189]: Super-User (root) logged in on port
/dev/pts/4 from host ccapollo.in2p3.fr.
Feb 20 17:20:22 ccpnds00 ntpd[5436]: stats: dc -0.279344 comp 0.000104
  peersw 83 inh 865 off -0.012649 SYNC 134.158.69.135 3
Feb 20 17:25:22 ccpnds00 ypxfrd [5436]: Cant'get master of services.
  Reason : no such map in server's domain
```

Pour tester l'ajout d'un événement dans le journal, il existe un petit utilitaire nommé : logger. Exemple :

```
logger -p mail.debug « Mon message »
```

Il est alors possible de visualiser l'évènement avec une commande :

```
tail /var/log/maillog
```

ou

```
tail -f /var/logmaillog
```

14 Trivial file Transfer Protocol (TFTP)

La commande *TFTP* permet le transfert de fichiers entre deux machines du réseau. Cette commande

est beaucoup moins complète que *FTP*, en particulier, elle n'utilise pas de système d'authentification, elle n'a pas de commande pour lister les fichiers distants, il n'est pas possible de changer de répertoire.

Bien que sommaire, *TFTP* est souvent utilisée par des machines réseaux, des terminaux X, etc, pour télécharger leurs systèmes d'exploitation, leurs fichiers de configuration, les polices de caractères ... Lors de problèmes de téléchargement de ces appareils, l'utilisation de la commande sur une machine UNIX permet dans bien des cas de déterminer les origines des problèmes et ainsi de les resoudre.

Sur la machine distante, un serveur *tftpd* doit être déclaré dans le fichier */etc/inetd.conf*. Ce server doit, de préférence (par mesure de securité), utiliser un compte permettant de restreindre les acces aux fichiers à une arborescence bien déterminée. Par exemple sur une machine *HP-UX* la ligne suivante du fichier */etc/inetd.conf* :

tftp dgram udp wait root /etc/tftpd -R 10 tftpd

permet d'utiliser *TFTP* avec les droits d'accés de l'utilisateur *tftp*. Cette déclaration peut être différente d'une version d'unix à l'autre, il est donc conseillé de consulter les manuels de la commande *tftp* et du fichier *inetd.conf*.

tftp utilise le port **69** et le protocol *udp*, sur beaucoup de sites, ce couple port/protocole est filtré en entrée et en sortie de site par mesure de securité.

La commande *tftp* peut être utilisée en mode interactif ou en mode commande. Dans le mode interactif, l'invite *tftp*> indique que l'on peut passer des commandes. En mode commande, celles-ci sont passées directement sur la ligne de commande et il n'y a pas d'invite.

Les principales commandes disponibles sous *tftp* sont :

connect node [port] permet d'indiquer la machine distante avec laquel seront effectués les transferts. *tftp* ne fonctionnant pas en mode connecté, cette commande ne crée pas de connexion.
get fichier-distant [fichier-local] permet de rapatrier un fichier distant sur la machine local. fichier-distant peut avoir la forme : `node:nom-du-fichier` où *node* est le nom de la machine distante, ce nom sera retenu pour les transferts suivants si *node* n'est plus spécifié.
put fichier-local [fichier-distant] permet de mettre un fichier local sur une machine distante. fichier-distant peut prendre la même forme que pour la commande get. Dans tout les cas, il faut avoir droit d'écriture sur le répertoire distant, dans certains cas (par mesure de securité), il faut que le fichier distant éxiste déja pour que le transfert soit possible.
ascii, binary indiquent dans quel type doit être éffectue le transfert, le défaut est ascii
quit permet de quitter le mode commande de tftp. CTRL/D a la même fonction.

Exemples :

mode interactif

```
$ tftp
tftp> connect snoopy
tftp> get /boot/os os
tftp> quit
$
```

mode commande

```
$ tftp get snoopy:/boot/os os
$
```

Ces deux exemples montrent comment rapatrier le fichier *os* de la machine *snoopy* sur la machine locale. Il faut noter que le répertoire */boot* et le fichier *os* doivent être en lecture pour les 'autres'.

Ce protocole peut servir aussi pour stocker la configuration d'appareils réseaux comme les commutateurs, les routeurs qui pratiquement tous savent rechercher ou écrire leur configuration ou leur image système sur des serveurs *TFTP*. C'est généralement par ce protocole que l'on met à jour les versions systèmes des matériels réseaux.

15 DHCP

DHCP signifie *Dynamic Host Configuration Protocol*
DHCP a pour objectif de fournir à des machines qui se connectent au réseau les informations de configuration nécessaire. *DHCP* peut en particulier fournir :

- Une adresse IP pour la machine qui fait la requête
- Le nom de domaine
- La liste des adresses des *DNS*
- Le masque de sous-réseau
- La passerelle par défaut
- Le nom du domaine *NIS*
- Le nom d'une image à télécharger
- ……

On peut considérer que *DHCP* remplace *BOOTP*.
La configuration de dhcp est effectuée par le fichier */etc/dhcpd.conf*. Si le fichier est modifié il faut redémarrer le service pour prendre en compte la modification. (/etc/init.d/dhcp restart)
L'intérêt de *DHCP* est qu'il peut distribuer des adresses IP dites **statiques** ou fixes, en fonction de l'adresse matérielle de la carte ethernet du client demandeur ; ou bien **dynamique**s c'est-à-dire prise dans un « pool » d'adresse. Ces adresses seront données « en location » c'est-à-dire pour un bail (durée) donné. Le client devra renouveler ce bail donc redemander une attribution avant la fin de l'expiration de la précédente.
Dans la pratique pour des adresses dynamiques le serveur vérifie si le client a déjà obtenu une adresse précédemment et lui reattribue si possible la même. Sauf si entre temps pour cause de pénurie il l'a attribuée à une autre machine. Le serveur tient pour cela à jour un fichier des locations attribuées dans */var/dhcp/dhcpd.leases*. Selon les versions il faut créer ce fichier manuellement avant le premier démarrage de dhcp.
L'intérêt de DHCP est :

- De permettre de n'avoir aucune configuration locale à faire sur les clients (dhcp est en général le défaut proposé par Linux)
- De ce fait de garantir l'unicité des adresses IP
- Avec les adresses dynamiques de pouvoir accueillir des visiteurs sur le réseau sans aucune formalité
- Au contraire en mode statique uniquement, et combiné avec des mécanismes liés aux matériels réseau (blocage des adresses *MAC, VLAN*…..), de pouvoir éviter toute intrusion physique sur le réseau
- Permettre de cloner des machines sans se soucier de la duplication d'adresse

Exemple de fichier de configuration *dhcpd.conf*

```
# dhcpd.conf
#
# IPNL configuration file for ISC dhcpd
# option definitions common to all supported networks...
ddns-updates off;
ddns-updates-style none;
authoritative;

option domain-name "in2p3.fr";
option domain-name-servers 134.158.138.50, 134.158.69.104, 134.158.69.191;
default-lease-time 600;
max-lease-time 7200;

  # reseau formation
  subnet 134.158.82.64 netmask 255.255.255.224 {
        option subnet-mask 255.255.255.224;
        option broadcast-address 134.158.82.95;
        #option nis-domain formation;
        #option nis-servers lyofors01;
        option routers 134.158.82.65;
        range  134.158.82.90 134.158.82.94;
  }

  # declaration pour une machine
  host lyoac12  {
        hardware ethernet 00:90:c2:c9:06:c0;
        fixed-address 134.158.142.18;
  }
```

Example de fichier */etc/dhcpd.leases :*

```
lease 134.158.82.90 {
    starts 2 2005/05/17 17:03:40;
    ends 2 2005/05/17 17:13:40;
    hardware ethernet 00:0c:db:a5:e4:21;
    uid 01:00:0c:db:a5:e4:21;
    client-hostname "ccpl";
}
lease 134.158.82.94 {
    starts 5 2005/05/13 14:55:32;
    ends 5 2005/05/13 15:05:32;
    hardware ethernet 00:0c:dc:0a:90:66;
}
```

16 Network Time Protocol (NTP)

NTP est un service qui à l'aide d'un serveur **ntpd** ou **xntpd**, lancé au démarrage de la machine, synchronise la date de la machine à partir d'une source externe située sur le réseau internet. La synchronisation s'effectue, en tenant compte des délais de propagation sur le réseau, en faisant "glisser" l'horloge de la machine ; certains systèmes ne prévoit pas d'appel système pour effectuer ce glissement *(HPUX* par exemple), il faut donc, dans ces cas là, lancer un serveur supplémentaire (*adjtimed*) avant de démarrer *ntpd*.

Les informations nécessaires à *ntpd* sont placées dans le fichier **/etc/ntp.conf**, les principales intructions que l'on doit mettre sont :

peer adresse-du-serveur [version] indique que la date de la machine locale va se synchroniser sur la date fournie par la machine dont l'adresse est indiquée, mais qu'en revanche, si nécessaire, la machine distante va se synchroniser sur votre date. Version est egale a 1 ou 2, par défaut c'est 1 pour communiquer avec des serveurs *ntpd*, 2 pour *xntpd*.

server adresse-du-serveur [version] indique que la date de la machine locale va se synchroniser sur la date fournie par la machine dont l'adresse est indiquée.

broadcast adresse indique à la machine locale qu'elle doit émettre des paquets NTP sur cette adresse. C'est en général l'adresse de "broadcast" de votre réseau (ou sous-réseau).

broadcastclient yes indique que la machine doit écouter les paquets *NTP* sur l'adresse de "broadcast". Dans ce cas c'est un client *NTP*.

driftfile fichier indique au serveur *NTP* d'enregistrer les écarts d'horloge dans fichier. Si ce fichier existe au démarrage, l'écart qu'il contient sera utilisé. Ce fichier est généralement */etc/ntp.drift*.

17 Le service de noms : DNS

17.1 Fonctionnement

Le service de noms (*Domain Name System*) est défini par les RFC 1032, 1033, 1034 et 1035. C'est un système mondial, coopératif, cohérent et hiérarchisé de nommage. La gestion des informations de la base de données globale est décentralisée. L'usage général est indépendant des types d'applications et du type de machines : du micro au mainframe !

- Les équipements communiquent grâce à leur adresse IP.
- Seules les applications utilisent les noms des équipements
- A une adresse IP peut correspondre un ou plusieurs noms (alias)

Un nom doit être unique au monde. Exemple :

@Ip : 134.158.138.13 = lyopcs9.in2p3.fr (alias lyoserv.in2p3.fr)

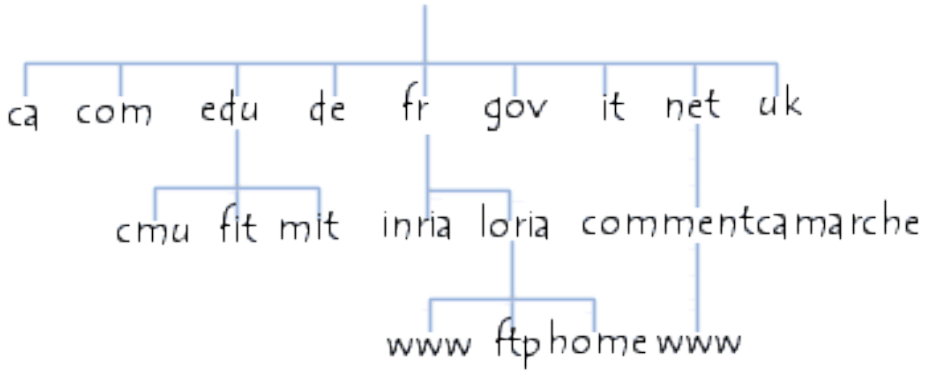
case insensitive (majuscule ou minuscule indifférent)

Il y a deux mécanismes distincts de résolution de noms :

- Résolution normale : nom -> adresse(s) ?
- Résolution inverse : adresse -> nom(s) ?

Le rôle du service de noms est la transformation des noms mnémoniques en adresses réseau et inversement. Les adresses des domaines de plus haut niveau sont gérées par le *NIC* (*Network Information Center*), qui a délégué son autorité à d'autres organismes pour gérer les sous-domaines correspondants. En France, c'est *l'INRIA* qui gére le domaine "fr", mais il a lui-même dans la plupart des cas délégué son autorité dans chaque organisme constituant l'ensemble du sous-domaine "fr", par exemple, c'est l'unité réseaux du *CNRS* (*UREC*) qui gére "cnrs.fr". Dans le jargon DNS, on parle

plutôt de 'zones'.



Il n'y a donc pas une base de donnée unique et centralisée pour l'ensemble des machines connectées à l'internet, mais plutôt, une multitude de bases de données (au moins une par organisme connecté). C'est le *DNS* (*Domain Name Service*) qui effectue les mises à jours et l'interrogation de ces données.

Les serveurs primaires détiennent l'origine des informations d'une zone (ensemble de machines) grâce à des fichiers de configuration

/etc/named.conf

/etc/named.data/db.hosts

/etc/named.data/named.local

Les serveurs secondaires (esclaves) obtiennent leurs informations (périodiquement ou à la demande) par un échange (transfert de zone) avec le serveur primaire

Les clients : demandent une résolution de nom (via un "resolver" *(gethostbyname, gethostbyaddress)*)

Sur chaque machine utilisant le service des noms, le fichier **/etc/resolv.conf** doit contenir le nom du domaine à laquelle elle appartient, ansi que les adresses des serveurs de noms qu'elle doit utiliser.

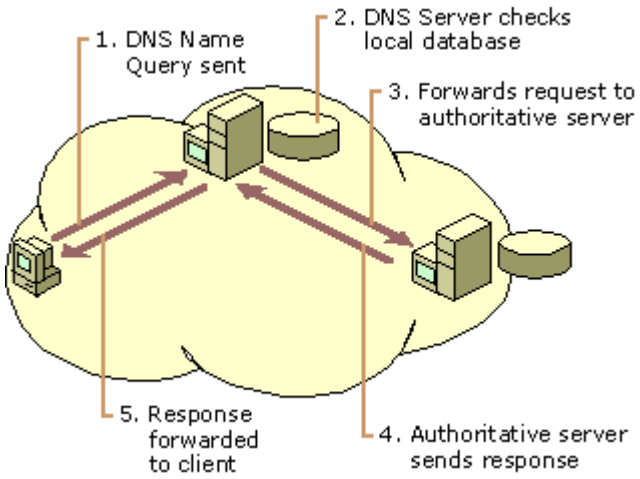
Exemple de fichier *resolv.conf*

```
;Fichier resolv.conf de la machine snoopy.fictif.fr
;cette machine est serveur secondaire du domaine
domain    fictif.fr
nameserver  127.0.0.1
nameserver  193.248.200.10
```

Si ce fichier est présent sur une machine, alors, les commandes de la forme :

`% telnet machine` seront équivalentes à `% telnet machine.fictif.fr`;

et pour toute résolution de noms, dans le domaine *fictif.fr* ou ailleurs, on fera appel à l'un des serveurs dont l'adresse est 127.0.0.1 (adresse de bouclage interne) ou 193.248.200.10.



Dessin 4: LEs requètes DNS

Requêtes itératives :

- Le serveur retourne soit la bonne réponse soit une référence vers un autre serveur de nom à contacter
- Les serveurs non récursifs répondent seulement aux requêtes pour les zones pour lesquels ils font "autorité". Ils ne conservent pas d'information en cache !

Requêtes récursives :

- Les serveurs récursifs ont obligation de donner réponse au client. Ils questionnent à leur tour d'autres serveurs jusqu'à obtention de la réponse (ou erreur)...leur cache se remplit

17.2 Les serveurs

Serveur primaire, serveurs secondaires, serveurs caches
Pour chaque zone, on peut distinguer 3 types de serveurs :

Le ’serveur primaire d'une zone’ est la source des informations contenue dans une zone. Ces informations sont stockées dans des fichiers de configurations générés soit par saisie manuelle (à l'éditeur de texte) soit par extraction d'une base de données. La réalisation de ces fichiers de configurations n'est pas du ressort du *DNS*. Dans sa zone, le serveur primaire fait autorité incontestable sur les informations de la zone. Il n'y a qu'un seul serveur primaire par zone.

'Les serveurs secondaires d'une zone' sont des serveurs qui obtiennent les informations de la zone à partir du serveur primaire. La liste des serveurs secondaires d'une zone est précisée dans la description de la zone elle-même. A intervalles réguliers (indiqués dans la description de la zone), les serveurs secondaires d'une zone transfèrent les informations de la zone, du serveur primaire vers leurs fichiers de configurations : c'est le rafraîchissement de la zone. Les serveurs secondaires font aussi autorité sur la zone. Il est à noter que les délais entre rafraîchissements peuvent provoquer des incohérences momentanées entre les informations du serveur primaire et les informations des serveurs secondaires. En cas de redémarrage du serveur secondaire, celui-ci <<rechargera>> les informations de la zone à partir de ces fichiers de configurations, puis éventuellement, si le délai est dépassé, effectuera un rafraîchissement de la zone (si le serveur primaire est accessible).
Tous les serveurs de noms sont également 'serveur cache' de toutes zones. En effet, ils conservent en mémoire toutes informations déjà demandées. Ce cache fait qu'au bout de quelques heures de fonctionnement, le serveur de nom d'un organisme contient en mémoire l'essentiel des informations concernant les zones avec lesquelles il communique. Il n'y a donc pratiquement plus de requêtes

DNS vers l'extérieur.

Il est à noter qu'un serveur primaire d'une zone peut aussi être serveur secondaire pour d'autres zones.
Réalisation du service des noms
La fonction de serveur de noms est réalisée à l'aide d'une tâche de fond, de deux fichiers de configuration et des fichiers de données exportées par le serveur.

La commande *named* : Cette commande est généralement située dans les répertoires */etc, /usr/etc* ou bien */sbin*, elle peut parfois être nommée **in.named**. Elle est lancée en tâche de fond, en précisant le nom de son fichier d'initialisation, au moment du démarrage du système. C'est cette tâche qui répond aux requêtes de résolution de noms et d'adresses pouvant provenir de machines internes ou externes a la zone. Si le serveur est serveur secondaire, la commande *named* fait appel à la commande **named-xfer** ou **in.named.xfer** pour effectuer les transferts d'informations depuis le ou les serveurs primaires. Lorsque *named* a fini son initialisation, elle écrit son PID dans un fichier */etc/named.pid*. Il est possible de forcer *named* à relire ses fichiers de configuration et de données avec un signal *SIGHUP*, par exemple avec la commande :

kill -HUP `cat /etc/named.pid`

17.3 Les fichiers de configurations

Le fichier d'initialisation, généralement appelé /**etc/named.conf**, contient les informations nécessaires au démarrage de la tâche *named*, soit :

```
// permit-transfer : les 2 serveurs de la zone in2p3 du CC
//
acl permit-transfer {
        134.158.69.104;
        134.158.69.191;
};
//
options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        # denis : puiseque ce serveur a plusieurs adresses IP
        # il faut que le notify provienne depuis l'@IP officielle
        # déclarée auprès des slave
        notify-source 134.158.138.50;
        // autoriser les transferts depuis les serveurs du CC
        allow-transfer {
                { permit-transfer; } ;
        } ;
};
//
// reduire la verbosite pour ce qui n'est pas de notre controle
//
logging {
        channel "my_channel" {
            syslog local5;
            severity warning;
            print-category yes;
            print-severity yes;
        };

        category "default" { "my_channel"; "default_debug"; };
        category "resolver" { null; };
};
```

```
//
// Serveurs racines
//
zone "." {
        type hint;
        file "/var/named/named.root";
};
//
// Cette commande permet de definir que le serveur est primaire pour
// le reverse mapping de 127.0.0.0
//
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "/var/named/master/named.in2p3local";
};
//
// Zone directe in2p3.fr nous sommes slave
//
zone "in2p3.fr" {
        type slave;
        file "/var/named/slaves/named.in2p3data";
        masters {
                134.158.69.191;
        };
};
//
// Zone inverse in2p3.fr nous sommes slave
//
zone "158.134.IN-ADDR.ARPA" {
        type slave;
        file "/var/named/slaves/named.in2p3rev_134_158";
        masters {
                134.158.69.191;
        };
};
//
// Zone ipnl.in2p3.fr nous sommes master
//
zone "ipnl.in2p3.fr" {
        type master;
        file "/var/named/master/ipnl.db";
        notify yes;
        # denis also-notify => notify yes;
        # also-notify {
        #       134.158.69.104;
        #       134.158.69.191;
        # };
};
```

La liste des zones gérées en tant que serveur primaire et/ou serveur secondaire. Un serveur primaire est au moins serveur primaire de trois zones, sa zone directe, sa zone inverse et la zone inverse du réseau de bouclage interne `127.0.0.0.` La zone directe associe un nom de machine ou de domaine avec un certain nombre d'informations (adresse réseau, relais de messagerie, serveurs de noms...); la zone inverse associe une adresse IP à un nom de machine. Un serveur secondaire est serveur primaire de sa zone inverse du réseau de bouclage interne et d'au moins deux zones, une zone directe et la zone inverse correspondante.
Le nom du fichier cache contenant les coordonnées des serveurs de la racine. Il est généralement nommé *root.cache*, *named.root* ou *named.ca* et contient le nom et les adresses IP des serveurs de la racine ("*root nameservers*"). Ces serveurs gérent la liste des serveurs de noms des zones nationales ou multinationales et la zone inverse *in-addr.arpa*. Ce fichier quel que soit son nom doit avoir le même contenu sur tous les DNS, il est disponible à l'URL ftp:// ftp.rs.internic.net/domain/root.cache et change trés peu souvent (moins d'une fois par an).
éventuellement des clauses modifiant le comportement standard du serveur (*forwarders*, *slave* ...).

Exemple de fichier *named.cache*

```
;    This file holds the information on root name servers needed to
;    initialize cache of Internet domain name servers
;    (e.g. reference this file in the "cache . <file>"
;    configuration file of BIND domain name servers).
;
;    This file is made available by InterNIC registration services
;    under anonymous FTP as
;      file        /domain/named.root
;      on server      FTP.RS.INTERNIC.NET
;    -OR- under Gopher at  RS.INTERNIC.NET
;      under menu     InterNIC Registration Services (NSI)
;       submenu     InterNIC Registration Archives
;      file        named.root
;
;    last update:  Nov 8, 1995
;    related version of root zone:  1995110800
;
;
; formerly NS.INTERNIC.NET
;
.            3600000 IN NS  A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.   3600000   A   198.41.0.4
;
; formerly NS1.ISI.EDU
;
.            3600000   NS  B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.   3600000   A   128.9.0.107
;
; formerly C.PSI.NET
;
.            3600000   NS  C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.   3600000   A   192.33.4.12
;
; formerly TERP.UMD.EDU
;
.            3600000   NS  D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.   3600000   A   128.8.10.90
;
; formerly NS.NASA.GOV
;
.            3600000   NS  E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.   3600000   A   192.203.230.10
;
; formerly NS.ISC.ORG
;
.            3600000   NS  F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.   3600000   A   192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.            3600000   NS  G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.   3600000   A   192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.            3600000   NS  H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.   3600000   A   128.63.2.53
;
; formerly NIC.NORDU.NET
;
.            3600000   NS  I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.   3600000   A   192.36.148.17
; End of File
```

17.4 Les fichiers de description de zones

Dans ces fichiers, les informations sont organisées en enregistrements appelés **Resources Records** (**RR**).

Type	Signification	Valeur
SOA	Start of Authority(acte de naissance)	Paramètres de la zone
A	Adresse IP d'u hote	Entier sur 32 bits
MX	Relais de messagerie	Domaine prenant le courrier
NS	nom de serveur	Nom d'un serveur pour ce domaine
CNAME	Nom de canonique	Nom de domaine
PTR	Pointeur	Alias pour une adresse IP
FO	Description de l'hote	UC et Système d'exploitation en ASCII
TXT	Texte	Texte ASCII non interprété

La forme des *resources records* est :

{nom} {tdv} classe-d'adresse type-du-record donnees-specifiques-au-record

Le second paramètre est un temps de vie (optionel).

Le caractère @ dans le champ nom indique l'origine courante, alors que la directive $ORIGIN commençant obligatoirement en colonne 1 permet d'indiquer un changement d'origine pour la suite des instructions du fichier. Chaque fois qu'un nom de machine apparaît dans un champ, l'origine est concaténée à la fin du nom, pour supprimer cette fonctionalité il faut mettre un point "." à la fin du nom.

La directive $INCLUDE suivie d'un nom de fichier, permet d'inclure celui-ci pendant la lecture. Cela permet d'organiser les données d'une zone primaire en plusieurs fichiers séparés.

Les retours à la ligne ne sont pas possibles, sauf s'ils apparaissent dans un groupe entouré de parenthèses.

Il y a trois types d'informations dans ces fichiers :

les informations administratives concernant la zone (enregistrement de type *SOA*),
la liste des serveurs primaires et secondaires de la zone (enregistrement de type *NS*),
puis les données elles-mêmes (enregistrement de type A, *CNAME* ou *MX* pour les zones directes, *PTR* pour les zones inverses).

L'enregistrement de type *SOA*

Les données spécifiques a cet enregistrement sont :

- le nom de la machine serveur primaire de la zone, ici *ns-primaire.fictif.fr*
- l'adresse électronique de la personne qui gère la zone (il faut noter que dans cette adresse @
- est remplacée par un point), ici *postmaster@snoopy.fictif.fr*
- puis 5 champs destinés aux serveurs secondaires qui sont dans l'ordre ou ils doivent apparaître :
  - un numéro de série qui doit toujours être croissant, c'est ce numéro qui permet à un serveur secondaire de savoir s'il doit faire appel à *named.xfer* lorsque le délai de rafraîchissement est arrivé a terme (il est donc conseillé d'utiliser une syntaxe de la forme *YYMMDDHH*) ;
  - la fréquence de rafraîchissement en secondes ;
  - la fréquence des re-essais en secondes ;
  - la durée de vie avant expiration des données (dans le cas de perte de connexion avec le serveur primaire) ;
  - la durée de vie minimale.

Exemple d'enregistrement *SOA*

```
@   IN  SOA  ns-primaire.fictif.fr. postmaster.snoopy.fictif.fr. (
        2009062501 ; Serial YYYYMMDDnn
        3600       ; refresh   1h
        600        ; retry    10min
        604800     ; expire    7j
        86400      ; minimum   1j
      )
```

Il ne doit exister qu'un seul enregistrement de type *SOA* pour une zone.

L'enregistrement de type *NS*

Cet enregistrement ne comprend qu'une seule donnée spécifique : un nom de serveur (primaire ou secondaire) de la *zone*. Il doit y avoir plusieurs enregistrements de type *NS* dans une zone (au moins deux pour des machine dans la zone et deux hors de la zone).

Exemples d'enregistrements *NS*

```
    IN  NS ns-primaire.fictif.fr.
    IN  NS snoopy.fictif.fr.
```

L'enregistrement de type A

Dans cet enregistrement, le nom est le nom de la machine à qui ont veut affecter l'adresse (qui est la donnée spécifique de l'enregistrement). Si une machine a plus d'une adresse, il n'est pas nécessaire de remettre le nom de la machine sur les lignes suivantes.

Exemples d'enregistrements A

```
ns-primaire  IN  A  193.248.200.10
       IN  A  193.160.128.233
```

L'enregistrement de type CNAME

Il fait la correspondance entre un alias et le vrai nom d'une machine.

Exemple d'enregistrement CNAME

```
licenceserv  IN  CNAME snoopy
```

L'enregistrement de type *MX*

Les enregistrements de type *MX* (*Mail eXchanger*) sont très importants pour la transmission du courrier électronique. Ils permettent entre autre, l'utilisation de machines relais de messagerie pouvant être des <<pare-feu>> ou des machines d'attente.

Les données spécifiques à cet enregistrement sont : la priorité et le 'mail exchanger'. Pour la priorité, plus le nombre est grand plus la priorité est faible.

Exemples d'enregistrements *MX*

```
snoopy  IN  MX  20  snoopy.fictif.fr.
     IN  MX  10  ns-primaire.fictif.fr.
```

L'enregistrement de type PTR

Ce type d'enregistrement n'apparaît que dans les fichiers de description de zone inverse. C'est lui qui permet de faire la correspondance entre une adresse IP et un nom de machine. Le nom indique l'adresse inverse de la machine (voir exemple), le champ données spécifiques comporte uniquement le nom de la machine.

Exemple d'enregistrement PTR

```
10.200.248.193.in-addr.arpa. IN  PTR  ns-primaire.fictif.fr
```

Exemples de fichiers de description de zones

17.4.1 hosts.db

C'est le fichier qui contient les enregistrements pour un serveur primaire

```
@   IN  SOA  ns-primaire.fictif.fr. postmaster.snoopy.fictif.fr. (
                    96061009
                    7200
                    1200
                    3600000
                    86400
                    )
;
       IN  NS   ns-primaire.fictif.fr.
       IN  NS   snoopy.fictif.fr.
;
localhost   IN  A   ns-primaire.fictif.fr.
       IN  MX   10 snoopy.fictif.fr.
       IN  MX   20 ns-primaire.fictif.fr.
```

```
ns-primaire  IN  A   193.248.200.10
       IN  MX   10 snoopy.fictif.fr.
       IN  MX   20 ns-primaire.fictif.fr.
snoopy    IN  A   193.248.200.150
       IN  MX   10 snoopy.fictif.fr.
       IN  MX   20 ns-primaire.fictif.fr.
licenceserv  IN  CNAME snoopy.fictif.fr.
surfer    IN  A   193.248.200.22
       IN  MX   10 snoopy.fictif.fr.
       IN  MX   20 ns-primaire.fictif.fr.
requin    IN  A   193.248.200.177
       IN  MX   10 snoopy.fictif.fr.
       IN  MX   20 ns-primaire.fictif.fr.
```

17.4.2 hosts.rev

C'est le fichier qui contient les enregistrements pour un serveur primaire, pour la zone inverse :

```
$ORIGIN in-addr.arpa.
200.248.193 IN SOA ns-primaire.fictif.fr. postmaster.snoopy.fictif.fr. (
                    96061009
                    7200
                    1200
                    3600000
                    86400
                    )
;
       IN  NS   ns-primaire.fictif.fr.
       IN  NS   snoopy.fictif.fr.
;
10.200.248.193  IN PTR ns-primaire.fictif.fr.
150.200.248.193 IN PTR snoopy.fictif.fr.
22.200.248.193  IN PTR surfer.fictif.fr.
177.200.248.193 IN PTR requin.fictif.fr.
```

17.4.3 named.local

Définit le reverse pour localhost

```
@  IN  SOA  ns-primaire.fictif.fr. postmaster.snoopy.fictif.fr. (
                    1
                    3600
                    300
                    3600000
                    3600
                    )
   IN  NS  ns-primaire.fictif.fr.
1  IN  PRT localhost.
```

17.5 La commande nslookup

La commande **nslookup** permet d'interroger un serveur de nom (ou le fichier *hosts* si le *DNS* n'est pas utilisé) pour obtenir des informations sur les machines connectées ou sur les zones, en fonction du type d'enregistrement sélectionné. Lorsque l'on active la commande les requêtes sont faites sur

les enregistrements de type A. Cette commande peut être utilisée en mode commande ou en mode interactif.

En mode commande, le nom de la machine dont on veut connaître l'adresse est passée en argument sur la ligne de commande suivie éventuellement d'un nom de *DNS* si l'on ne veut pas utiliser le défaut ; il est aussi possible de faire la résolution inverse. Pour passer les options du mode interactif il faut les faire précéder du signe moins (-).

Exemple en mode commande :

```
$ nslookup -query=MX snoopy
Server: ns-primaire.fictif.fr
Address: 193.248.200.10

snoopy.fictif.fr  preference 20, mail exchanger = ns-primaire.fictif.fr
snoopy.fictif.fr  preference 10, mail exchanger = snoopy.fictif.fr
fictif.fr     nameserver = ns-primaire.fictif.fr
fictif.fr     nameserver = snoopy.fictif.fr
ns-primaire    inet address = 193.248.200.10
snoopy       inet address = 193.248.200.150
$
```

Les principales options disponibles sous l'invite (>) du mode interactif de *nslookup* sont :

host [server] nom de la machine dont on veut connaître l'adresse suivie éventuellemet d'un nom de DNS.

server nom nom du serveur DNS à utiliser pour les requêtes suivantes.

set querytype=valeur valeur étant le type d'enregistrement pour les requêtes suivantes (A, NS, MX, SOA, CNAME,...).

set type=valeur synonyme de querytype.

ls [options] domain [> fichier] liste les informations pour la zoneindiquée par domain (éventuellement redirigé dans un fichier). L'option **-t** suivie du type d'enregistrement souhaité permet d'avoir la liste de ces enregistrements pour la zone indiquée.

exit ou CTRL/D permet de sortir de la commande.

Exemple en mode interactif :

```
$ nslookup
Default server: ns-primaire.fictif.fr
Address: 193.248.200.10

> snoopy
Server: ns-primaire.fictif.fr
Address: 193.248.200.10

Name:  snoopy.fictif.fr
Address: 193.248.200.250

> server jaguar.ulysse.fr
Default server: jaguar.ulysse.fr
```

```
Address: 196.52.105.2

> set querytype=SOA

> fictif.fr
fictif.fr origin = ns-primaire.fictif.fr
      mail addr = postmaster.snoopy.fictif.fr
      serial=96061009, refresh=7200, retry=1200, expire=3600000, min=86400

> fictif.fr
Non-authoritative answer:
fictif.fr origin = ns-primaire.fictif.fr
      mail addr = postmaster.snoopy.fictif.fr
      serial=96061009, refresh=7200, retry=1200, expire=3600000, min=86400

Authoritative answers can be found from:
fictif.fr nameserver = ns-primaire.fictif.fr
fictif.fr nameserver = snoopy.fictif.fr
ns-primaire    inet address = 193.248.200.10
snoopy       inet address = 193.248.200.150
```

> exit

17.6 La commande DIG

La commande dig est plus récente que nslookup, elle permet de donner des informations plus complètes :

La syntaxe de la commande dig est la suivante :

```
dig @<serveur> <domaine> <type_de_requete>
```

Affichage du *mail exchanger* du domaine lyon.fr :

```
$ dig @134.158.138.50 mx lyon.fr +short
100 mta.teaser.net.
200 mta1.teaser.net.
200 mta2.teaser.net.
```

Affichage des serveurs qui ont autorité sur le domaine ipnl.in2p3.fr :

```
$ dig @134.158.138.50 ns ipnl.in2p3.fr +short
lyodns.in2p3.fr.
ccpntc3.in2p3.fr.
ccpnvx.in2p3.fr.
```

Affichage de l'adresse IP d'une machine :

```
$ dig @134.158.138.50 a lyodns.in2p3.fr +short
134.158.138.50
```

Affichage du nom reverse correspondant à une adresse IP :

```
dig -x 134.158.138.50 +short
lyodns.in2p3.fr.
```

18 Network Information Service

NIS (autrefois connu sous le nom de **Yellow Page**) est un service mis au point par *SUN* pour administrer de façon centralisée un certain nombre de fichiers système. Ce mécanisme permet de partager sur plusieurs stations des fichiers de manière transparente. Les modifications ne se font

physiquement que sur une seule machine, mais sont ’propagées’ automatiquement vers toutes les autres.

Ainsi, les fichiers */etc/bootparams*, */etc/ethers*, */etc/group*, */etc/hosts*, */etc/netgroup, /etc/netid, / etc/netmasks, /etc/networks, /etc/passwd, /etc/protocols, /etc/publickey et /etc/services /etc/ aliases* peuvent etre gérés de cette manière. On peut bien sur ajouter ses propres fichiers a cette liste. Les fichiers sont propagés sous la forme de fichiers dbm (DataBase Manager)

Avantages
- La gestion est simplifiée. La mise à jour d'un fichier sur le serveur provoque la mise à jour de manière transparente sur toutes les machines du domaine.
- Un plus grand confort est offert aux utilisateurs, qui ne possèdent qu'un seul password quelque soit la machine du domaine sur laquelle ils se connectent. Changer le password sur une station le change sur toutes les autres.

Inconvénients :
- NIS ne fonctionne pas parfaitement en milieu hétérogène. Il existe quelques incompatibilités entre certains constructeurs.
- Il y a plus gênant. Si le serveur est indisponible les machines clientes sont bloquées jusqu'au redémarrage du serveur. Notamment on ne peut plus se connecter puisque le fichiers *passwd* est propagé par le serveur. Le réseau ne fonctionne plus si le serveur *NIS* distribue les adresses *IP* des hosts.

On utilise donc des serveurs secondaires, mais ils ne permettent pas de modifier les *maps* propagées.

Terminologie
L'ensemble des machines **serveur maitre**, **serveur(s) esclave(s)**, **clientes** constitue un domaine.

Le serveur maitre est la seule station a pouvoir effectuer des modifications Il possède le jeu de *map* maitre et le propage aux serveurs esclaves. Il répond aux clientes.

Les serveurs esclaves possèdent des copies des *maps* mais ne peuvent effectuer de modifications. Ils reçoivent leurs copies des *maps* du serveur maitre ou d'un autre serveur secondaire. Ils répondent aux clientes. Ils sont utilisés pour palier une défaillance du serveur maitre, ou pour décharger le serveur maitre.

Les clientes utilisent les services *NIS.*

18.1 Les maps

Les maps sont équivalentes aux fichiers de */etc* mais sous une forme différente.
Elle sont au format **dbm**. On a donc pour chaque map un fichier **map.dir** et un fichier **map.pag**. Ce sont des bases de données dans lesquelles les recherches s'effectuent au moyen d'une clef. Pour des raisons d'efficacité, chaque fichier est indexé avec plusieurs clefs. Ainsi le fichier *passwd* est indexé a la fois par nom d'utilisateur et par uid. On trouvera donc 4 fichiers pour cette map : p*asswd.byname.pag, passwd.byname.dir, passwd.byuid.pag* et *passwd.byuid.dir*.

Les maps résident dans le répertoire NIS */var/yp/domainname* Ces fichiers sont générées automatiquement a partir des fichiers de /etc par les commandes :

```
cd /var/yp
make
```

Mécanisme
Toutes les machines exécutent la commande *domainname*

Les serveurs exécutent la commande *ypserv* et *ypbind*.
Les serveurs sont en général clients d'eux même.
Le serveur maitre en plus exécute la commande *yppasswdd* et la commande *ypxfrd*.

Les clientes exécutent la commande ***ypbind***. Les clientes reçoivent les informations du serveur NIS grâce au binding, mis en œuvre par la commande *ypbind*. *ypbind* effectue un *broadcast* pour trouver un serveur de son domaine. A tout instant, si le serveur est indisponible, *ypbind* tente automatiquement de trouver un autre serveur par un nouveau *broadcast*.

Une fois NIS lancé, les fichiers suivants sont 'complétés' par NIS :
```
/etc/passwd
/etc/group
/etc/bootparams
/etc/hosts
/etc/aliases
```

Sur un client, de manière transparente, la recherche s'effectue d'abord par consultation des fichiers locaux de */etc* puis s'il y a lieu en interrogeant le serveur NIS. Ce fonctionnement dépends de l'ordre des directives « *files* » et « *nis* » dans le fichier */etc/nsswitch.conf* .

Les fichiers suivants sont remplacés:
```
/etc/ethers
/etc/netgroup
/etc/netmasks
/etc/networks
/etc/protocols
/etc/services
```

Changement pour l'administration : **Le mécanisme d'expiration des mots de passe ne fonctionne plus.**

Changement pour les utilisateurs : Le seul changement notable pour les utilisateurs se situe au niveau de la commande *passwd*. Cette commande ne gère pas *NIS*. Les utilisateurs doivent alors changer leur mot de passe avec la commande ***yppasswd***, qui s'emploie de la même manière que la commande *passwd*. Il en est de même pour les commande *chsh* et *chfn* qui ne fonctionnent plus ; elles sont remplacées par les commandes *ypchsh*, *ypchfn*

18.2 Mise en œuvre de NIS

Pour mettre en œuvre NIS il faut effectuer les opérations suivantes :
- choisir un nom de domaine identique sur toutes les machines
- configurer une machine serveur maitre
- configurer d'éventuels serveurs esclaves
- configurer les machines clientes

Le nom de domaine *NIS* : Il ne faut pas confondre le nom de domaine *NIS* et le nom de domaine *BIND* ou *DNS.* Le nom de domaine *DNS* est précisé dans le fichier */etc/resolv.conf* alors que le nom de domaine *NIS* l'est dans */etc/yp.conf*

exemple de fichier */etc/yp.conf* :
```
domain ardeche server nom_d_un_serveur_NIS
```

Le nom de domaine est fixé par la commande domainname. Par exemple

```
# domainname ardeche
```

Cette commande doit donc être lancée au démarrage par un */etc/init.d/ypbind* par exemple pour le cas d'un client NIS.
Si on observe le mécanisme de ces scripts, ils font référence au fichier **/etc/sysconfig/networks** à l'intérieur duquel est définie la variable ***NISDOMAIN***.

18.2.1 Configurer le serveur Maitre

Après avoir lancé la commande *domainname*, on doit construire les maps, déclarer les serveurs esclaves et lancer les daemon.

La commande **ypinit** permet de configurer les serveurs Pour configurer un serveur maitre la commande est **ypinit -m** (voir dans le répertoire /usr/lib/yp/)

exemple :
```
# ypinit -m
Installing the NIS data base will require that you answer a few questions.
Questions will all be asked at the begining of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n :n] n
Ok, please remember to go back and redo manually whatever fails. If you
don't, some part of the system (perhaps the NIS itself) won't work.

At this point, we have to construt a list of the hosts which will run NIS
servers. Please continue to add tha names for the others hosts, one per line.
When you are done with the list, type a <Control-D>.
  next host to add : ardechesud
next host to add : ^D

The current list of NIS server looks like this :

ardechesud

Is this correct? [y/n :y] y
There will be no further questions. The remainder of the procedure should take
5 to 10 minutes.
Building /var/yp/ardeche/ypservers...
Running /var/yp/Makefile...
updated passwd
updated group
updated hosts
updated ethers
updated networks
```

```
updated services
updated protocols
make : warning : don't know how to make target `/etc/netgroup'
Current working directory /var/yp
make : warning : don't know how to make target `/etc/bootparams'
Current working directory /var/yp
```

Pour lancer les daemons sur le serveur maitre, il faut faire :

```
# service ypserv start
# service yppasswdd start
# service ypxfrd start
# service ypbind start
```

Le *daemon* ***ypxfrd*** n'est pas obligatoire. Il accélère simplement les transferts de *maps* entre les serveurs secondaires et le serveur maitre.

La commande ***yppasswdd*** devant être lancée au démarrage, il faudra veiller à ce que le script */etc/init.d/yppasswdd* soit référencé dans le répertoire */etc/rc5.d* (ici on suppose que le niveau de démarrage est égal à 5)

18.2.2 Configurer les serveurs esclaves

La configuration des serveurs esclaves est très simple :

Lancer la commande *ypnit* (*ardechenord* est le nom du serveur maître) :
```
# ypnit -s ardechenord
```
Lancer le service *ypserv* :
```
# service ypserv start
```
Configurer le fichier client *ypbind* et lancer le client *ypbind* :
```
# service ypbind start
```

Pour être efficace, un tel système doit s'assurer que les *maps* sont identiques sur tous les serveurs. On peut synchroniser les serveurs de deux manières. Soit le serveur maitre informe les serveurs secondaires d'un changement dans une ou plusieurs *map*s soit les serveurs secondaires demandent des mises à jour des *maps* au serveur maitre.

Sur le serveur maitre, on re-propage une *map* par la commande ***yppush***
Exemple :

```
# yppush passwd
```

Si le fichier source de la *map* a change, le plus simple est de refaire un *make* de la *map* en question. Ainsi après avoir ajouté un utilisateur dans le fichier *passwd* du serveur maitre, il faut refaire un *make passwd* pour reconstruire la *map* et transférer les changement aux clients. exemple :

```
# cd /etc
# vi group
```
ajouter un groupe
ou un membre à un groupe
:wq

```
# adduser ...
```
ajouter les utilisateurs
```
# cd /var/yp
# make group
Updating group.byname...
Updating group.bygid...
ardechenord# make passwd
Updating passwd.byname...
Updating group.bygid...
```

Sur les serveurs secondaires, on demande la mise à jour des *maps* par la commande **ypxfr.**

```
# ypxfr -h ardechenord passwd.byname
Map passwd.byname at ardechenord is not more recent than local.
```

Il existe des scripts qui se chargent de recharger les maps sur les serveurs clients plusieurs fois par jour et qu'il faut lancer par des crontab. Il est judicieux de lancer ces scripts à des heures différentes sur chaque serveur secondaire afin de ne pas surcharger le serveur maitre au moment du transferts.

/usr/etc/yp/ypxfr_1perhour	transfère la map passwd
/usr/etc/yp/ypxfr_1perday	transfère les maps group protocols networks services ypservers.
/usr/etc/yp/ypxfr_2perday	transfère les maps hosts ethers netgroup aliases.

Exemple de crontab sur les serveurs scondaires:

```
0 8,20 * * * /usr/etc/yp/ypxfr_2perday
10 10  * * * /usr/etc/yp/ypxfr_1perday
15 1-23 * * * /usr/etc/yp/ypxfr_1perhour
```

18.2.3 Configurer les clientes

Ceci est valable sur toutes les machines sauf le serveur maitre.

Il faut :

- définir le nom du domaine : *ypdomain formation*
- modifier le fichier */etc/yp.conf*
  - *domain formation server serveur_nis_primaire*
  - *domain formation server serveur_nis_secondaire*
- modifier les scripts de configuration réseau (*/etc/sysconfig/network*) pour y indiquer le domaine :
  ```
  NISDOMAIN=formation
  ```
- (en option) : préparer les fichiers */etc/passwd* et */etc/group*. Avec Linux, on s'affranchit de cette étape.
- Modifier le fichier */etc/nsswitch.conf* pour définir les maps qui NIS qui devront être utilisées , par exemple :
  - `passwd:  files nis`

    - `shadow:   files nis`
    - `group:    files nis`
    - `netgroup: files nis`
    - `hosts:    files dns nis`
- lancer le démon *ypbind* via le service *ypbind* dans */etc/init.d :*
  commande : */etc/init.d/ypbind start*
  ou
  commande : *service ypbind start*

Préparation des fichiers de groupe et des utilisateurs
passwd:

Dans une configuration classique, on doit rajouter à la fin du fichier */etc/passwd* des entrées spéciales indiquant que la recherche doit se poursuivre en interrogeant NIS.

Ces entrées ont la forme suivante :

`+nom:`
précise que l'entrée nom des map NIS est valable sur cette machine cliente

`+nom::::home-dir:`
précise que nom est valable sur cette machine; mais le home-directory de nom est celui précise dans le fichier *passwd*

> Pour que cela fonctionne, il faut que le mot clé « *compat* » soit positionné à la place de « *nis* » dans le fichier */etc/nsswitch.conf* pour la *map passwd*.

`+::0:0:::`
indique que toute entrée de la map NIS est valable sur cette machine.

exemple de fichier *passwd* sur une machine cliente:

```
root:Ahj87dsddzG9d8z:0:3::/:/bin/sh
daemon:*:1:5::/:/bin/sh
bin:*:2:2::/bin:/bin/sh
adm:*:4:4::/usr/adm:/bin/sh
uucp:*:5:3::/usr/spool/uucppublic:/usr/lib/uucp/uucico
lp:*:9:7::/usr/spool/lp:/bin/sh
hpdb:*:27:1:ALLBASE:/:/bin/sh
+::0:0:::
```

group:

Le fichier group doit subir des modifications semblables

`+toto:`
indique que le groupe *toto* est valide sur cette machine

`+:`
toutes les entrees de la map nis *group* sont valables sur cette machine cliente

exemple de fichier group sur un client:

```
root::0:
other::1:
bin::2:
sys::3:
adm::4:
daemon::5:
mail::6:
lp::7:
+:
```

18.2.4 En résumé

Sur le serveur maître :

- ypserv est lancé
- ypbind est lancé
- rpc.yppasswdd est lancé
- ypxfrd est lancé

Sur le(s) serveur(s) esclaves :

- ypserv est lancé
- ypbind est lancé

Sur un client :

- Si on est en mode « *compat* »
  - */etc/passwd* contient a la fin une entrée du type +::0:0:::
  - */etc/group* contient a la fin une entrée du type +:
- ypbind lancé lors du démarrage (fichier de configuration /etc/yp.conf)

Sous Linux, l'utilisation du fichier **/etc/yp.conf** permet de s'affranchir d'un *broadcast* en nommant explicitement le serveur comme :

```
domainname ardeche server ardechenord
```

18.3 Les netgroups

Les netgroups ne sont disponibles que si *NIS* est présent sur la machine. Les netgroup permettent de définir des groupes de machines et des groupes d'utilisateurs. Ces noms de groupes peuvent alors être utilisés partout ou un nom de machine ou utilisateur est attendu. Ainsi on peut définir des netgroup que l'on utilise pour contrôler les *export NFS*, ou pour contrôler les *logins*. Toutefois, si on utilise les *netgroup* pour le fichier *passwd*, La configuration du serveur maitre doit légèrement changer. Nous verrons cela plus loin.

Qu'est ce qu'un **netgroup** ?

Un netgroup est un ensemble de triplet (*machine*, *utilisateur*, *NIS domain*) défini dans le fichier **/etc/netgroup**. Il ne faut pas, bien que cela soit techniquement possible, mélanger des utilisateurs et des machines dans un même groupe, le résultat pourrait ne pas être celui escompté. Dans un triplet (*machine, utilisateur, NIS domain*), l'absence d'une valeur est un joker. Un champ commençant par une valeur non alphanumérique comme le signe - indique que le champ est rendu inutilisable. Le

troisième champ n'a pas d'utilité pratique.

Le fichier */etc/netgroup* suivant :

```
ardecheu  (,christophe,) (,andre,) (,philippe,)
ardechem  (ardechenord,,) (ardechesud,,) (ardecheest,,)\
          (ardecheouest,,)
```

défini 2 netgroups, ardecheu qui est groupe d'utilisateur, et ardechem qui est un groupe de machines.

On peut a partir de là, donner le nom de groupe dans les fichiers :
/etc/hosts.equiv ~/.rhosts ou */etc/exports*

Par exemple le fichier */etc/exports* pourrait contenir une ligne du type :

```
/cdrom -access=ardechem -ro
```

Les fichiers passwd peuvent contenir des lignes du style : **+@ardecheu:** ce qui a pour effet d'inclure les membres du groupe ardecheu

ou bien encore

```
+@ardecheu::0:0:::/bin/noshell
```

ici les utilisateurs du groupe *ardecheu* sont toujours connus sur la station, mais leur *login shell* a été changé en un *shell* qui n'existe pas. les utilisateurs de ce groupe ne peuvent pas se logger.

A cause de la manière dont fonctionne *NIS*, si on utilise les *netgroup* pour le fichier *passwd*, sur le serveur maitre, le fichier */etc/passwd* ne peut plus servir à la fois de source pour les *map* et de fichier *passwd*. En effet les lignes +... ne sont pas consultées, car on trouve dans une ligne antérieure l'entrée que l'on recherche. La solution consiste donc a utiliser un autre fichier que */etc/passwd* pour construire la map. On peut par exemple copier */etc/passwd* dans /etc/passwd.nis, ôter les entrée locales telles que *root, uucp, lp, bin*, etc... et ne laisser que les entrées devant être diffusées par *NIS*. Il ne faut pas oublier de modifier le fichier */var/yp/Makefile*, sans quoi on construirait une map ne contenant que les entrées *root, bin, lp* etc... On peut aussi créer un répertoire distinct /etc/nis, copier tous les fichiers de */etc* servant aux map dans ce fichier. De la même manière il faut éliminer les entrées locales, dans */etc/nis/passwd*, changer la commande *.yppasswdd* pour indiquer le fichier a changer, modifier la macro *DIR* dans le fichier */var/yp/Makefile*. Exemples :

fichier */etc/nis/passwd*

```
christophe:VonKKghnXS8aGs:100:60:Christophe Martin:/home/martin:/bin/tcsh
alain:JNRAdF1Vnv.Gt6:101:60:Alain Durand:/home/alain:/bin/csh
andre:X2uezn9s979nCM:153:150:Andre Dupont:/home/bisquer:/bin/tcsh
```

fichier */etc/passwd*

```
root:Ahj8sdf7dG9d8z:0:3::/:/bin/sh
daemon:*:1:5::/:/bin/sh
bin:*:2:2::/bin:/bin/sh
adm:*:4:4::/usr/adm:/bin/sh
```

```
uucp:*:5:3::/usr/spool/uucppublic:/usr/lib/uucp/uucico
lp:*:9:7::/usr/spool/lp:/bin/sh
hpdb:*:27:1:ALLBASE:/:/bin/sh
+@ardeche:
+::0:0:::
```

fichier */var/yp/Makefile :*

```
B=
DIR =/etc/nis         # cette ligne était autrefois DIR=/etc
DOM = `domainname`
NOPUSH = ""
ALIASES = /etc/aliases
YPDIR=/usr/etc/yp
YPDBDIR=/var/yp
YPPUSH=$(YPDIR)/yppush
MAKEDBM=$(YPDIR)/makedbm
```

18.4 Les commandes NIS

Outre le partage de fichiers transparent, *NIS* offre en plus la possibilité de questionner directement un serveur au moyen de certaines commandes.

ypwhich. *ypwhich* donne le nom du serveur ou indique s'il y a un problème Exemple :

```
% ypwhich
can't clntupd_create: Can't communicate with ypbind
```

Diagnostic: *ypbind* n'est pas lancé sur cette machine

```
% ypwhich
ardechenord
```

ypwhich peut aussi indiquer la liste des alias pour les maps connues

```
% ypwhich -x
Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr"
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
Use "ethers" for map "ethers.byname"
```

ypcat. *ypcat* affiche toute la map spécifiée:

```
% ypcat protocols
udp   17   UDP   # user datagram protocol
pup   12   PUP   # PARC universal packet protocol
tcp   6    TCP   # transmission control protocol
ggp   3    GGP   # gateway-gateway protocol
```

```
igmp  2    IGMP  # internet group multicast protocol
icmp  1    ICMP  # internet control message protocol
ip   0    IP   # internet protocol, pseudo protocol number
```

l'option **-t** indique à *ypcat* de ne pas de considérer le nom de la map comme un alias. Exemple :

```
% ypcat -t ethers
no such map in server's domain
% ypcat ethers
8:0:20:b:28:f  ardechenord
8:0:20:b:20:b  ardechesud
8:0:20:b:27:e4 ardecheest
8:0:20:3:9f:85 ardecheouest
```

ypmatch. *ypmatch* recherche les clefs dans une map :

```
%ypmatch uucp passwd
uucp:*:4:8::/var/spool/uucppublic:
```

yppoll. *yppoll* indique le nom de domain et le serveur maitre pour une map donnée:

```
% /usr/etc/yp/yppoll passwd.byname
Domain ipnl is supported.
Map passwd.byname has order number 834997399.
The master server is ardechenord
```

19 Network File System

Du point de vue logique, lorsqu'un utilisateur se trouve connecté à un système Unix, il observe dans tous les cas une structure de fichiers arborescente à partir de la racine /.

Du point de vue physique, les fichiers vus par l'utilisateur peuvent être soit **locaux**, c'est à dire résidants sur un disque physique attaché à la machine, soit **distants**, c'est à dire sur un périphérique ne disposant pas d'un attachement matériel direct sur la machine. Dans ce dernier cas, on parle de **système de fichiers distribué**.

NFS (*Network File System*) est certainement un des systèmes de fichiers distribués des plus répendus qui offre aux utilisateurs un accès transparent à l'ensemble de ses fichiers sans avoir besoin de savoir où résident physiquement ses fichiers. *DFS* (*Distributed File System*), *AFS* (*Andrew File System*) sont d'autres systèmes de fichiers distribués mais dont les mécanismes internes sont très différents.

Pour l'administrateur système, *NFS* permettra de centraliser la gestion de l'espace disque, par exemple en offrant à chaque utilisateur un répertoire home unique à travers le réseau ou encore en diffusant un ensemble d'utilitaires qui seront gérés depuis un répertoire unique.

NFS est typiquement une application client/serveur :
Les machines qui mettent à disposition des fichiers sur le réseau sont dites serveuses, on parle alors d'exportation de fichiers;
Les machines qui utilisent ces fichiers sont les clientes. Elles ont accès à des fichiers résidants dans

des répertoires ou des systèmes de fichiers qui préalablement ont été montés.

NFS est indépendant des constructeurs et des architectures réseau. Il utilise le protocole ***RPC*** (*Remote Procedure Call*) pour faire communiquer un serveur et un client. Un client émet une requête RPC pour information en direction du serveur qui lui renvoie le résultat.

NFS s'appuie sur deux protocoles internes distincts : **MOUNT** et **NFS**; le premier étant utilisé par le serveur pour déterminer quels systèmes de fichiers sont disponibles et à destination de quelle machine; le second pour mettre les fichiers à disposition du client.

Ce sont des paquets de 32 octets appelés "*file handle*" qui sont échangés entre serveur et client pour contrôler l'accès aux fichiers. 18 appels RPC sont utilisés pour réguler l'échange de données entre serveur et client, il s'agit de :

null.	Permet d'effectuer des tests de disponibilité (c'est le "ping RPC")
gettattr.	Renvoie les attributs d'un fichier
setattr.	Définit les attributs d'un fichier
root.	Teste les permissions "root" auprès du serveur
lookup.	Ouvre un fichier et un retourne ses attributs et un "file handle" au client
readlink.	Lit un lien symbolique
read.	Lit un fichier
wrcache.	Appel au cache lecture/écriture.
write.	Ecrit dans un fichier
create.	Crée un fichier
remove.	Détruit un fichier
rename.	Renomme un fichier
link.	Crée un lien matériel sur un fichier
symlink.	Crée un lien symbolique sur un fichier
mkdir.	Crée un répertoire
rmdir.	Supprime un répertoire
readdir.	Lit un répertoire
fsstat.	Renvoie l'état d'un système de fichiers.

Pour garantir cette indépendance constructeur, les transactions NFS doivent reposer sur un protocole d'échange standardisé de données, il s'agit de ***XDR*** (*eXternal Data Representation*). Aussi, si un serveur n'utilise pas le même ordonnancement des octets pour le stockage des données qu'un client, *XDR* prendra en charge la conversion vers un format standard.

Au niveau des couches de transport, *NFS* peut utiliser aussi bien les protocoles *TCP qu'UDP*. *UDP* est rapide mais non fiable, c'est à dire qu'il n'y a aucune garantie qu'un paquet émis soit délivré correctement ou qu'une suite de paquets arrivent dans l'ordre d'émission. Pour contourner ce problème, lorsque le serveur *NFS* répond à un appel *RPC*, il l'accompagne d'un code de retour pour l'action qu'il a exécutée. De son coté, si le client ne reçoit pas de réponse à une requête, il la retransmet pour l'action qu'il a exécutée.

Aucune manipulation spécifique n'est nécessaire pour pouvoir utiliser *NFS*. Lors d'un appel à une primitive de manipulation de fichiers, le système d'exploitation détermine si le fichier est local ou distant et génère l'appel adéquat :

- un appel aux programmes d'entrées/sorties standard;
- un appel *RPC* (*Remote procedure Call*) dans le cas de données distantes.

La mise en œuvre de *NFS* sur un réseau de machines *Unix* se fait en démarrant un certain nombre de démons sur les machines clientes et serveuses. Viennent en complément, des fichiers de configuration exploités par ces démons et par les procédures d'initialisation du système.

Parmi ces démons, il en existe un qui va permettre à une application utilisant des *RPC* (ici en l'occurrence *NFS*) d'identifier sans ambiguïté le canal de communication entre un client et un serveur, il s'agit de **portmap**.

Par définition, on associe à une application *RPC*, un numéro de programme et un numéro de version. Le client qui connait ces valeurs ainsi que le nom du serveur, devra également connaitre le numéro de port associé au protocole *UDP* afin de pouvoir communiquer avec le serveur.
C'est la fonction de *portmap* que de permettre à un client d'accéder au numéro de port sur lequel un serveur pourra accepter ses requêtes.
Lorsqu'un serveur *NFS* démarre, il s'enregistre auprès du démon *portmap* qui tourne sur le même système et maintient une table des ports associés aux applications s'y exécutant. Quand au client NFS, il contacte son démon *portmap* local qui lui rendra le port sur lequel dialoguer avec le serveur. Toutes les requêtes ultérieures, utiliseront directement ce même port.

Le serveur *NFS*
Le serveur *NFS* a pour fonction, en réponse à des requêtes de machines clientes *NFS*, de lire ou d'écrire des fichiers sur ses disques locaux.

Il ne maintient cependant pas d'information sur les fichiers ouverts par des machines clientes pour la bonne raison qu'en cas d'incident sur le serveur, la reprise en est simplifiée. Ce sera donc au client d'effectuer cette tâche; le client devra ré-émettre sa dernière requête jusqu'à ce que le serveur redevienne disponible.

19.1 Démarrage d'un serveur

On configure une machine en serveur *NFS* en lançant les démons **nfsd** et **rpc.mountd** dans cet ordre.
nfsd ou rpc.nfsd gère les opérations sur les systèmes de fichier (ouverture, lecture, écriture, etc...). Un processus nfsd ne sait traiter qu'une seule transaction à la fois, c'est pourquoi il est d'usage de le démarrer comme :
`# nfsd n` (*n* étant un entier, **8** une valeur par défaut raisonnable)

On peut aussi utiliser les scripts de démarrage des services réseaux dans /etc/init.d/nfs :

```
# /etc/init.d/nfs start
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
```

en conséquence, *n* démons *nfsd* seront démarrés, le processus *nfsd* initial effectuant un appel système *fork()* pour démarrer les *n-1* autres "démons fils" *nfsd*.

rpc.mountd a pour fonction de répondre aux requêtes de montage d'un client NFS en fonction des informations stockées dans le fichier /etc/xtab. Il maintient également une liste des montages NFS actifs sur les machines clientes.

Seules les machines clientes autorisées par le serveur pourront accéder aux disques du serveur. La mise à disposition de ces ressources disque se fera grâce au mécanisme d'exportation de systèmes de fichier ou de répertoires.

19.2 L'exportation

C'est le fichier **/etc/exports** qui décrit quels systèmes de fichiers ou répertoires du serveur pourront être accessibles et par quelles machines clientes. Chaque ligne de ce fichier est constituée par le chemin absolu du système de fichier ou de la portion d'arborescence à exporter suivi d'une liste de machines clientes auxquelles s'appliquent des autorisations.

Ces autorisations sont les suivantes :

- *ro* L'exportation se fera en lecture seule. Le client ne pourra accéder aux fichiers qu'en lecture même si le montage a été effectué en lecture/écriture. Toute tentative d'écriture depuis un client se traduira par un message du type "Permission denied". C'est le défaut.
- *rw* Définit explicitement la liste des machines autorisées à accéder en écriture au système de fichiers. Si *rw* est rencontré seul, tous les clients NFS seront autorisés en écriture.
- *no_root_squash* Définit la liste des machines où l'utilisateur *root* sera à même de modifier les fichiers du système de fichier. Par défaut, *root* ne dispose pas de privilège particulier contrairement au cas où il accède à un système de fichiers local.
- *anonuid* Si au niveau de la machine cliente NFS, le propriétaire d'un fichier n'est pas défini (ou est défini comme l'utilisateur *anonymous*), il se verra réaffecter une nouvelle identification *uid*.
- *All_squash* chaque uid ou gid est mis en correspondance avec ceux de l'utilisateur *anonymous*

Examinons l'exemple suivant :

```
/home host1(rw),host2
/utils/scripts host1(rw),host2(rw),mainthost(rw,no_root-squash)
/usr/local/bin (ro)
```

Les systèmes de fichiers */home* xet */utils/scripts* seront accessibles depuis les machines *host1* et *host2*. Néanmoins *host2* n'accède à */home* qu'en lecture seule. Le super-utilisateur *root* de la machine *mainthost* pourra modifier */utils/scripts*. Quant au système de fichiers */usr/local/bin,* il sera accessible en lecture seule depuis n'importe quelle machine.

L'exportation d'un système de fichier ne sera effective qu'après exécution de la commande **exportfs**.

L'exécution de cette commande a pour effet de maintenir dans le fichier **/etc/xtab**, l'ensemble des enregistrements du fichier */etc/exports* qui sont présentement exportés. Le format du fichier */etc/xtab* est identique à celui de */etc/exports*.

La commande *exportfs* est généralement activée en fin de phase de boot sur la machine serveuse NFS et n'est exécutable que par le super-utilisateur *root* :

```
# exportfs -a
```

Dans ce cas, *exportfs* lit le contenu du fichier */etc/exports* et effectue autant d'exportations que le fichier contient de lignes.

Si l'on souhaite annuler les exportations actives, on utilisera l'option **-au** de *exportfs* :

```
# exportfs -au
```

Plus aucune transaction NFS depuis un client ne sera acceptée.

Pour connaitre les systèmes de fichiers exportés à un instant donné, exécuter la commande *exportfs* sans argument :

```
# exportfs
/home -access=host1:host2
/utils/scripts -access=host1:host2,root=mainthost
```

19.3 Le client NFS

Un client *NFS* accède à des systèmes de fichiers préalablement exportés par un serveur *NFS*. Le système de fichiers NFS (tout comme le système de fichiers *Unix*) ne sera utilisable qu'après une opération de **montage**.

Qu'il s'agisse d'un système de fichiers local ou *NFS,* une opération de montage revient à accrocher une autre arborescence en un point donné de l'arborescence locale.

Configuration d'un c*l*ient *NFS*

Dans le meilleur des cas, il n'y a rien à faire, le noyau gère intègre l'aiguillage en cas d'une demande d'accès à un fichier qui ne serait pas situé dans un système de fichiers local. C'est la cas pour les versions actuelles de Linux
Pour configurer, un système en tant que client *NFS*, il faudra préalablement démarrer les démons qui auront en charge d'effectuer les opérations de lecture/écriture sur les fichiers résidant sur le serveur. Ces démons appelés **nfsiod** ou **biod** (block I/O daemons) sont en exécution en permanence et sont généralement démarrés pendant la phase d'initialisation du système (classiquement dans les fichiers */etc/rc*)*. La syntaxe d'appel aux *biod* est similaire à celle des démons *nfsd* :

```
# biod 8
```
c'est à dire 7 démons *biod* fils du *biod* initial)

Toute opération de montage sera postérieure au démarrage des *biod*.

19.4 Le montage

Il existe 3 types de montage :
le montage explicite. Il s'agit d'un montage réalisé par le biais de la commande mount directement émise depuis le shell courant par le super-utilisateur root.
le montage prédéfini. C'est un montage dont la définition est décrite dans le fichier :

- **/etc/fstab** (systèmes Sun0S V4.x, Dec Unix);
- **/etc/filesystems** (système IBM AIX 3.x, 4.x);
- **/etc/checklist** (système HPUX 8.0x et 9.0x).

Lors de la phase d'initialisation d'un système qui opère en tant que client NFS, ces fichiers seront exploités par la commande **mount** pour effectuer l'ensemble des montages NFS décrits. Dans ce cas, la commande *mount* sera généralement appelée comme :

```
# mount -a -t nfs
```

qui monte tous les systèmes de fichiers (option **-a**) dont le type de montage est *NFS* (**-t nfs**).

l'auto-montage. Il s'agit d'un montage dynamique de répertoires ou de systèmes de fichiers.

La commande *mount ,* outre les options utilisées pour le montage de systèmes de fichiers locaux, admet un certain nombre d'options s'appliquant aux systèmes de fichiers distants. La syntaxe est la suivante :

```
# mount -t nfs [-o op1,op2,...] serveur:/rep-distant /rep-local
```

On précise le nom du serveur *NFS* à utiliser (ici *serveur*) séparé par le caractère "**:**" du nom du répertoire ou système de fichiers distant (*/rep-distant*) suivi du nom du répertoire d'encrage dans l'arborescence locale (*/rep-local*). Si ce dernier répertoire n'existe pas ou si l'exportation du répertoire distant n'est pas correcte, le montage échouera.

D'éventuelles options de montage *NFS* peuvent être ajoutées, séparées par des virgules, derrière l'option **-o** de *mount* . Il s'agit de :

ro, rw	Le répertoire sera monté respectivement en lecture seule ou en lecture/écriture
hard	On relance une requête de montage jusqu'à ce que le serveur réponde
soft	Si le serveur ne répond pas, on abandonne le montage et *mount* retourne un code d'erreur
bg	Si la première tentative de montage échoue, la tentative suivante sera effectuée en arrière plan
intr	Autorise les interruptions clavier lors d'un montage effectué avec l'option *hard*
retry=*val*	effectuera *val* fois la tentative de montage
rsize=*val*	Fixe la valeur des buffers de lecture à *val* octets
wsize=*val*	Fixe la valeur des buffers d'écriture à *val* octets

Quelques commentaires concernant ces options s'imposent. Tout d'abord les options *soft* et *hard* définissent les actions prises par le client lorsqu'un système de fichiers devient indisponible.
Dans le cas de l'option *hard* , si une entrée/sortie est en cours, elle sera retransmise indéfiniment, même si le nombre maximum de retransmissions est atteint. Si l'option *intr* n'a pas été spécifiée, l'application restera pendue jusqu'à ce que le système de fichiers réapparaisse. On rencontre typiquement ce problème, lorsqu'une station cliente *NFS* reboote et que le serveur est indisponible; on ne peut alors même plus se connecter...

Au vu des problèmes de montage *hard*, l'option *soft* n'est-elle pas la solution ? Effectivement, c'est bien adapté au cas où le montage est en lecture seule (option *ro*). Cependant lorqu'on est en lecture/écriture, abandonner en cours d'écriture, risque de se traduire par un fichier corrompu.

En résumé, nous préfèrerons les options *hard* et *intr* pour un montage *rw* (l'utilisateur reste alors maître de décider s'il veut interrompre son application et l'option *soft* pour un montage *ro*.

Commandes informatives coté client

Un client NFS pourra accéder à un certain nombre d'informations telles que les montages actifs ou bien les montages potentiels à partir d'un serveur donné, les caractéristiques de ces montages.

La commande **mount** sans argument liste les montages actifs.

```
# mount
/ on /dev/dsk/c201d6s0 read/write on Fri Jan 28 20:21:46 2005
/home on serveur:/home read/write on Thu Jan 27 11:38:36 2005
```

Dans cet exemple (extrait d'un système sous HP-UX), on observe deux sytèmes de fichiers montés :

- un système de fichiers local / monté sur un disque d'adresse SCSI 6 (c201d**6**s0)
- un système de fichiers NFS */home* exporté depuis la machine *serveur* et monté sous */home.*

La consultation de la liste des clients qui ont monté un système de fichiers et des attributs à l'exportation de systèmes de fichiers d'un serveur donné se fera par la commande **showmount**.

```
# showmount serveur
host1
host2
```

Les machines *host1, host2* sont des clientes *NFS* du serveur *xporter*. Ces informations sont extraites du fichier **/etc/**rmtab sur le serveur et sont mises à jour par le démon **rpc.mountd**.

Attention, il est possible de voir apparaitre dans cette liste, des machines qui ne sont pas présentes dans le fichier d'exportation */etc/exports*. Ceci se produit lorsqu'un client a été supprimé à l'exportation coté serveur et qu'il a disparu de façon "violente" du réseau (crash ou arrêt sans démontage explicite)

Utilisée avec l'option **-a**, *showmount* affichera quels répertoires d'un serveur donné sont montés sur quelles machines clientes :

```
# showmount -a serveur
host1:/home
host1:/utils/scripts
host2:/usr/local/bin
```

et avec l'options **-d**, ce seront les répertoires affectés par un montage qui seront affichés :

```
# showmount -d serveur
/home
/utils/scripts
/usr/local/bin
```

19.5 Protections des fichiers sous NFS et sécurisation de NFS

Le système de fichiers *NFS* ainsi que les systèmes de fichiers *Unix* locaux suivent les mêmes règles concernant la protection et l'accès aux fichiers. Autrement dit, 3 principaux modes d'accès aux fichiers (lecture, écriture et exécution) s'appliquant à 3 domaines d'utilisateurs (le propriétaire, le

groupe et les autres).

Les droits d'accès sont basés uniquement sur les *UID/GID* (un fichier du serveur dont *UID=150*, sera vu avec un *UID* de 150 sur la machine cliente) aussi faut-il s'assurer de l'unicité des *UID* entre serveur et machines clientes pour éviter les conflits. On peut donc aisément comprendre que l'on ne doit pas offrir un service *NFS* à des machines dont l'administrateur n'est pas un collaborateur clairement identifié ou de confiance.

D'autre part, il est très important de parfaitement maitriser la portée de l'exportation de systèmes de fichiers surtout lorsqu'elle est faite en lecture/écriture. Limiter l'exportation aux seules machines qui en ont besoin ou utiliser les *netgroups* de *NIS* sont des règles à respecter.

Une exportation rigoureuse sera garante d'une bonne configuration *NFS*, aussi faudra t-il que le fichier */etc/exports* soit parfaitement configuré avant de démarrer le serveur. Après toute modification de ce fichier, il faut absolument re-exporter.

Par défaut, *NFS* change *l'UID* 0 à -2 (utilisateur *nobody*) lorsque les requêtes proviennent de l'utilisateur *root* d'une machine cliente. Ceci n'est plus vrai si on utilise l'option *root* lors de l'exportation, ce qui permet alors à un utilisateur *root* distant de créer des fichiers (dans le système de fichier monté); ces fichiers devenant également la propriété de *root* sur le serveur. Donc dans la mesure du possible, il est préférable d'éviter l'option *root*.

Enfin, il faut éviter d'exporter des exécutables du serveur autrement qu'en lecture seule, c'est un moyen supplémentaire de protection du serveur *NFS*.

19.6 Statistiques d'utilisation de NFS

19.6.1 La commande netstat

Utilisée avec l'option -s, la commande **netstat** montre les statistiques par protocole :

```
netstat -s
udp:
    0 incomplete headers
    0 bad data length fields
    0 bad checksums
    0 socket buffer overflows
tcp:
    7357038 packets sent
        ....
```

Si *udp* signale des "socket buffer overflows", ceci provient de processus utilisateurs qui ne libèrent pas assez rapidement les sockets ouverts lors de transactions *NFS*. Il faut dans ce cas, pour satisfaire la demande, augmenter le nombre de démons serveurs. Par exemple, redémarrer *nfsd* sur le serveur comme :

nfsd 12

19.6.2 La commande nfsstat

La commande **nfsstat** permet d'obtenir des statistiques sur l'utilisation de *NFS* et de déterminer d'éventuels problèmes relatifs aux divers appels *RPC* de *NFS*. *nfsstat* s'utilise aussi bien coté serveur que coté client.

Coté serveur :
On utilise la commande *nfsstat* (option *-ns* : *n* signifie NFS et *s* serveur) :

```
nfsstat -ns
Server nfs:
calls   badcalls
148     0
null     getattr  setattr  root     lookup   readlink  read
0 0%    50 33%   0 0%    0 0%    54 36%   4 2%    37 25%
wrcache  write    create   remove   rename   link     symlink
0 0%    0 0%    0 0%    0 0%    0 0%    0 0%    0 0%
mkdir    rmdir    readdir  fsstat
0 0%    0 0%    0 0%    3 2%
```

Affiche le nombre d'appels *NFS* reçus, ceux rejetés (*badcalls*) et leur répartition entre les 18 appels *RPC/NFS*. En cas de problème de performances, il faut surveiller:

- *readlink* (maximum 10%). Au delà, les clients effectuent une utilisation excessive des liens symboliques, auquel cas, il faudra remplacer ces liens par des répertoires et donc effectuer de nouveaux montages
- *getattr* (maximum 60%). Normalement un client *NFS* maintient en cache les attributs de fichiers pendant un certain laps de temps sans être obligé à chaque fois de les redemander au serveur. Si cette valeur est dépassé, cela indique que les attributs de cache ont été réduits ou le cache supprimé.

Coté client :
On utilise toujours *nfsstat* (option *-rc* : *r* signifie *RPC* et *c* client) :

```
nfsstat -rc
Client rpc:
calls      retrans  authrefrsh
10724      4541     0
```

où les champs essentiels sont :
retans est le nombre d'appels retransmis. S'il est élévé les réponses à des requêtes au serveur n'ont pu revenir avant le *timeout*. On peut ajuster ça en modifiant la valeur de *retrans* de la commande *mount*.

20 L'administration système

20.1 Outils de diagnostic des processus

Pour connaitre tous les processus en cours de fonctionnement

- **ps** : liste les processus actifs lancés dans la console courante.
- **ps aux** : affiche la liste de tous les processus, avec leur numéro PID, le terminal tty où ils

ont été lancés (sinon ?). Voici la liste des colonnes du tableau obtenu .

- "USER" à quel utilisateur appartient le processus.
- "PID" est le numéro qui identifie le processus
- "%CPU" en % les ressources du microprocesseur utilisées par le processus.
- "%MEM" en % les ressources en mémoire vive utilisées par le processus.
- "RSS" mémoire réellement utilisée en ko par le processus.
- "START" l'heure à laquelle le processus a été lancé.

La commande *top* permet d'avoir une liste mise à jour toutes les quelques secondes de tous les processus. On peut trier :

- M : tri par mémoire
- P : tri par CPU
- N : tri par PID
- T : tri par temps processeur

Selection des processus

- 1 : affiche tous les processeurs
- i : affiche les tâches runnables
- k : tue un process
- r : renice

La commande lsof permet de voir toutes les ressources utilisées par un processus :

- *lsof fichier* : Voir tous les processus qui ouvrent un fichier
- *lsof repertoire* : Voir tous les processus qui ouvrent un sous répertoire
- *lsof -pPID* : Voir tous les fichiers ouverts par le processus PID
- *lsof -i @host* : Voir tous les processus ayant une connexion ouverte avec host
- *lsof -i TCP:port* : Voir tous les processus ayant une connexion tcp ouverte sur le port port
- *fuser -vm <device|montage>* : Voir tous les processus ouvrant des fichiers sur un device ou un point de montage

20.2 Outils d'analyse de performance

- *uptime* : affiche les « load average » des 1, 5, 15 dernières minutes
- *cat /proc/loadavg* : idem
- *vmstat* : affiche les taux d'utilisation de la RAM, SWAP
- *mpstat* : affiche la charge des processeurs
- *iostat* : affiche les statistiques des entrées / sorties
- *sar* : permet de collecter des informations sur le système à intervalles réguliers. Ces informations sont stockées sur le système et peuvent être analysées à posteriori :

```
sar option -s debut -e fin
-b : entrées sorties                -B : paging
-c : création de process            -d : activité sur les devices block
-n : activité réseau                -P : activité processeur
-q : load average                   -r : utilisation de la mémoire
-R : statistiques mémoire           -u : utilisation du CPU
-v : utilisation des inodes         -y : activité des tty
```

`sar -x PID 0 1` : monitorer en temps réel l'activité d'un processus

21 La sécurité sous UNIX

Lorsqu'il fut crée dans les années 70, le système *UNIX* était destiné à une utilisation interne chez *AT&T* et ses concepteurs n'avaient pas beaucoup de soucis de sécurité : Unix avait été créé par des développeurs pour des développeurs. Son évolution vers une solution "industrielle" comme système V ou encore la version universitaire *BSD* qui intégrait une ouverture vers les réseaux locaux et étendus, n'a pas pour autant résolu tous les problèmes initiaux. C'est pourquoi aujourd'hui encore, bien que de gros efforts aient été réalisés, on trouve encore dans les systèmes *UNIX* quelque soit le fournisseur, quelques éléments mettant en danger la sécurité du système.

Néanmoins, le gros des problèmes que l'on rencontre dans les systèmes *UNIX* est lié à des configurations ou à des mises en œuvre du système insuffisamment rigoureuses. Ceci est d'autant plus critique qu'aujourd'hui nous travaillons dans en réseau où prime l'ouverture et donc où la faiblesse d'un seul élément constitue un risque potentiel pour le reste du réseau.

On a souvent tendance à assimiler la sécurité informatique à la protection contre les accès illégaux à un système, ce n'est là que la partie émergée de l'iceberg. En réalité, la sécurité consiste à protéger l'information stockée sur les systèmes d'informatiques. Elle vise donc à assurer pour ces systèmes et pour les échanges d'informations dans lesquels ils sont impliqués :

- la disponibilité : un utilisateur autorisé désirant utiliser un système doit être servi dans les meilleurs délais;
- l'intégrité : l'information stockée sur le système se doit d'être fiable et modifiable uniquement par les utilisateurs disposant des autorisations adéquates (qu'il s'agisse des données ou des programmes);
- la confidentialité : pour une information donnée, seules les personnes autorisées à la connaître pourront y accéder.
- la non répudiation : il est indispensable que le destinataire d'une information valide ne puisse nier l'avoir reçue.

Gestionnaires de systèmes ou utilisateurs, la sécurité est l'affaire de tous. D'un coté, l'administrateur système se devra de :

- configurer les procédures d'identification et d'authentification;
- gérer des listes de contrôle d'accès;
- d'effectuer la sensibilisation nécessaire auprès des utilisateurs;
- assurer le suivi et l'audit du système et des utilisateurs.

Quand à l'utilisateur final, il devra également se protéger. Mais pourquoi?

- *pour sa propre activité*. Il pourra ainsi d'une part valider sans équivoque ses rapports, ses données, ses applications et d'autre part accéder confortablement aux ressources informatiques disponibles : CPU, disques, etc...
- *pour ses collaborateurs directs*. Il fera souvent parti d'un projet commun à plusieurs utilisateurs qui en cas de problème seraient eux aussi impactés
- *pour ses collaborateurs directs*. Il fera souvent parti d'un projet commun à plusieurs utilisateurs qui en cas de problème seraient eux aussi impactés
- *pour la communauté scientifique*. L'irresponsabilité d'un seul individu peut jeter le discrédit sur l'organisation à laquelle il appartient.

Administrateur de machines *Unix* vous êtes responsables du suivi de votre parc et donc de la sécurité de celui ci. En cas de problème, **avertissez immédiatement votre hiérarchie fonctionnelle**.

L'ensemble des points développés dans ce document ne constituent pas une liste exhaustive qui vous permettra de vous garantir à coup sûr contre tout incident mais plutôt de vous offrir quelques bases indispensables pour assurer un suivi régulier de votre parc.

L'identification et l'authentification
Ce sont les fichiers */etc/passwd* et */etc/group* qui sont les garants d'une authentification valide des utilisateurs du système :

- il doivent posséder un mode d'accès de **444** et appartenir à *root*;
- le second champ (séparé par :) ne doit pas être vide;
- dans la mesure du possible, utiliser le mécanisme de "*shadow password*" : le champ 2 du fichier *passwd* (le mot de passe encrypté) est déplacé vers un fichier non consultable par l'utilisateur lambda.

21.1 Protection des fichiers et commandes associées

Tous les fichiers de votre système doivent avoir un propriétaire opérant dans un groupe bien identifié, on peut vérifier que c'est bien le cas par :

```
find / \( -nouser -o -nogroup \) -exec ls -l {} \;
```

Vérifiez régulièrement l'absence de fichiers *SUID* root dans des systèmes de fichiers autre que ceux purement système :

```
find / -user root -perm 4000 -exec ls -al {} \;
```

Également, il faut s'efforcer de détecter les concentrations de fichiers disposant de protection faibles, le masque de création de fichier en est souvent le responsable.

La commande : **`umask 077`**
créera une valeur du masque par défaut acceptable. Si l'utilisateur en est l'origine, responsabilisez le, mais ne modifiez pas les droits sans son accord.

A propos du super utilisateur (root) : L'utilisateur root est l'utilisateur critique des systèmes Unix dans la mesure où il peut tout faire. Voici quelques recommandations :

- éliminez "." (répertoire courant) du chemin de recherche (*PATH*) de *root*;
- **vérifiez régulièrement l'historique des commandes de *root* et parcourez les fichiers de** *sulog* ou *syslog* où sont consignés entre autre le résultat d'une commande su ou d'une connexion;
- **Évitez de vous connecter** *root* directement par *rlogin* ou *telnet*. Certains systèmes permettent de n'autoriser que certains ports physiques de connexion (*/etc/ttytab* ou */etc/ttys* ou */etc/securetty* selon le système).

21.2 Les fichiers cachés et réseaux

.rhosts Est un fichier qui peut être présent dans le répertoire *home* d'un utilisateur. Il définit quels utilisateurs d'un système distant sont autorisés à utiliser des commandes du type "r-commandes" au nom de l'utilisateur local. Il est constitué par un ensemble de lignes :

système_distant utilisateur distant.

Par exemple :

```
machine1.orga.fr    tartempion
machine2.orga.fr
```

L'utilisateur *tartempion* sur *machine1.orga.fr* est autorisé sur la machine locale ainsi que tous les utilisateurs de *machine2.orga.fr*.
Ce fichier doit être protégé en lecture contre tout utilisateur autre que son propriétaire(mode **600**).

.netrc Ce fichier permet d'automatiser une session **ftp** ou **rexec**. Ce fichier peut contenir des mots de passe et devra donc être protégé en lecture tout comme .rhosts (mode **600**). Une entrée type permettant l'ouverture d'une session ressemble à :
machine mac1.org.fr login toto password $CouVert

/etc/host.equiv Est un fichier à n'utiliser qu'en cas de stricte nécessité. Il définit l'ensemble des systèmes distants autorisés à exécuter des commandes sur le système local sans être obligé de fournir de mot de passe. Il s'applique à tous les utilisateurs du système.
Si ce fichier contient une ligne avec un + (ce qui signifie tous les systèmes), la supprimer impérativement .

.forward Ce fichier qui permet de rediriger des messages de type "mail" ne doit contenir que des adresses électroniques. Ce fichier comme les précédents doit être protégé contre les opérations de lecture d'autres utilisateurs.
Les services réseau

Si l'on souhaite mettre en place des contrôles d'accès au niveau du réseau, on peut le faire à divers niveaux :

- au niveau des applications (démons) sur les stations *Unix*;
- dans les tables de routage de la station;
- en effectuant un filtrage dans les routeur du réseau;
- en utilisant des pare-feu (fire-wall) point de passage obligé du trafic venant du monde extérieur vers le réseau local et qui effectueront des tâche de contrôle d'accès, d'authentification ou même de comptabilité.

Nous n'aborderons dans ce document que le premier point où bon nombre de services réseau sont sous le contrôle du démon **inetd** (telnet, ftp, etc...).Ces services devront être déclarés dans le fichier */etc/inetd.conf* afin de pouvoir répondre aux requêtes de machines distantes. Certains de ces services peuvent être exploités à des fins d'intrusion, aussi il ne faut pas hésiter à supprimer un service que l'on utilise pas. Pour ce faire, insérer le caractère # sur la ligne concernée.

Le maintient des fonctionnalités suivantes doit être étudié en fonction de vos spécificités :

- **tftpd** Permet un transfert sans contrôle d'accès. Son utilisation ne se justifie réellement que sur des terminaux X ou des stations sans disque dans un cadre de téléchargement logiciel.
- **R-commandes** Il s'agit des démons dont le nom commence par la lettre "r" qui permettent de faire exécuter des tâche soumises à distance : *rlogind, rshd, rexecd*. Le risque

augmente avec l'affranchissement des phase d'authentification si les fichiers /*etc/host.equiv* ou *~/.rhost* existent.

21.3 La sécurité du système

Un système est composé d'une multitude d'utilitaires utilisables (pour leur majorité) par l'ensemble des utilisateurs du système. Un système *Unix,* grâce à ses capacités native de communication et à sa stabilité, est généralement utilisé pour offrir des services réseaux. Un système qui est connecté à Internet est donc accessible à distance par l'intermédiaire du réseau.

Il est bien connu qu'aucun logiciel n'est exempt de bug, ou d'erreurs de programmation. Dans certains cas, ces erreurs provoquent un plantage du logiciel en question, ce qui en conséquence provoque l'arrêt brutal du service rendu par ce logiciel. Dans d'autres cas, les bugs peuvent être exploités par un utilisateur local ou distant (à travers le réseau) pour que l'utilisateur obtienne un accès non prévu par le développeur. Cela va de l'accès non autorisé jusqu'à l'élévation de privilège (l'utilisateur obtient les privilèges de l'utilisateur root).

La plupart des éditeurs de distributions Linux incluent un mécanisme de mise à jour des logiciels inclus dans la distributions. Dans l'exploitation d'un système Linux connecté au réseau, il faut se préoccuper de la mise à jour des logiciels installés, c'est à dire :
- s'informer des vulnérabilités des logiciels installés sur la distribution Linux,
- mettre à jour les logiciels vulnérables dès que qu'elles ont été corrigées par les développeurs,

Le processus de mise à jour est particulièrement important pour les systèmes accessibles depuis tout l'Internet. Actuellement, nous ne sommes plus entre chercheurs ou ingénieurs œuvrant dans le domaine académique, il faut tenir compte du fait que l'Internet est devenu potentiellement hostile, vu depuis l'intérieur d'un laboratoire de recherche.

21.4 Le filtrage réseau

Les distributions Linux sont généralement équipées d'un système de filtrage intégré. Le plus connu et utilisé est *iptables* (ancien *ipchains*).

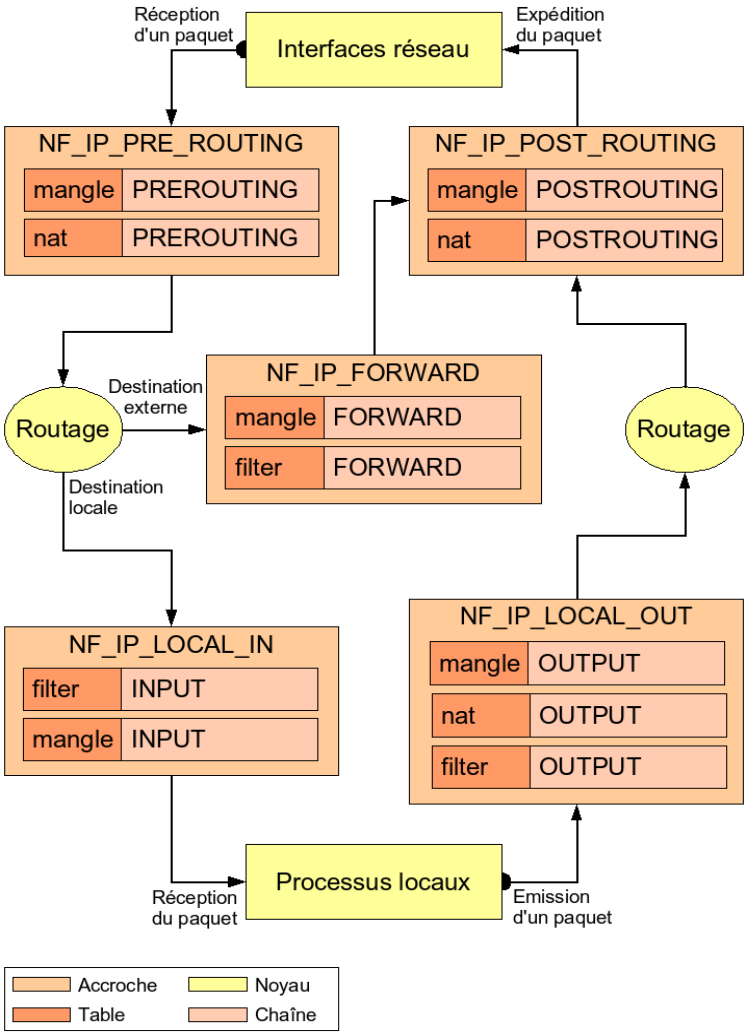


Illustration 16: Le filtrage avec IPTABLES

En premier lieu, lorsqu'un paquet arrive sur la machine où fonctionne un noyau 2.4, une décision de routage est prise :

- si le paquet n'est pas destiné à un processus local, il passe dans la chaîne `FORWARD` si le routage est activé, est détruit sinon ;
- si le paquet est destiné à un processus local, il passe dans la chaîne `INPUT` ;

Lorsqu'un processus local émet un paquet, celui-ci passe dans la chaîne `OUTPUT`.

Une décision est prise pour chaque paquet : *ACCEPT*, *DROP*, *QUEUE* ou *RETURN*.

- ***ACCEPT*** signifie que le paquet est autorisé à passer.
- ***DROP*** signifie que le paquet est détruit.
- ***REJECT*** est utilisé pour répondre à un paquet qui doit être stoppé, on réponds avec un paquet ICMP spécifié par l'option –reject-with :
  - cmp-net-unreachable (réseau inaccessible),
  - icmp-host-unreachable (machine inaccessible),
  - icmp-port-unreachable (port inaccessible),
  - icmp-proto-unreachable (protocole non utilisable),
  - icmp-net-prohibited (réseau interdit),
  - icmp-host-prohibited (machine interdite)

  - icmp-admin-prohibited (communication interdite par l'administrateur
- ***QUEUE*** signifie que le paquet est transmis à l'espace utilisateur (si cette option est reconnue par le noyau).
- ***RETURN*** signifie que l'on cesse de parcourir cette chaine pour retourner dans la chaîne précédente (appelante) en passant à la règle suivante. Si on atteint la fin d'une chaîne prédéfinie ou s'il y a correspondance avec une règle dans une chaîne prédéfinie ayant pour cible *RETURN*, la cible désignée par le comportement par défaut de la chaîne détermine le sort du paquet.

Sur les systèmes de type *Redhat*, le fichier qui contient les règles de filtrage sont dans */etc/sysconfig/iptables* :

```
cat /etc/sysconfig/iptables

# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
# syslog remote
-A RH-Firewall-1-INPUT -p udp -m udp --dport 514 -j ACCEPT
# pour Xnest
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 6001 -j ACCEPT
# snmp
-A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp --dport 161 -j ACCEPT
# ntp
-A RH-Firewall-1-INPUT -p udp -m udp --sport 123 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 123 -j ACCEPT
# cups
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 631 -j ACCEPT
# pour cfengine
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 5308 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp --dport 5308 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 5038 -j ACCEPT
# pour iperf en reception
# pour netcat en reception
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 10123 -j ACCEPT
# pour netcat en reception
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 10124 -j ACCEPT
# on loggue le reste
-A RH-Firewall-1-INPUT -j LOG --log-level info
# fin
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT

# service iptables
Usage: /etc/init.d/iptables {start|stop|restart|condrestart|status|panic|save}
```

Remarques :

Le "!" peut être utilisé pour certaines commandes afin de spécifier le contraire (on peut le traduire par "sauf"). Par exemple une commande qui doit refuser tout trafic TCP sauf ce qui provient de l'adresse IP 10.42.42.42 sera traduite par la commande suivante :

Exemple :

```
# iptables -A INPUT -p tcp --source ! 10.42.42.42 -j DROP
```

Les adresses IP peuvent optionnellement être spécifiées avec le masque associé sous la forme [adresse ip]/[masque].

`--state` : Permet de spécifier l'état du paquet à matcher parmi les états suivants :
`ESTABLISHED` : paquet associé à une connexion déjà établie
`NEW` : paquet demandant une nouvelle connexion
`INVALID` : paquet associé à une connexion inconnue
`RELATED` : Nouvelle connexion mais liée, idéal pour les connexions FTP

Exemples :
```
# iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-j ACCEPT
# iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j
ACCEPT
```

`-j (jump)` : Défini l'action à prendre si un paquet répond aux critères de cette règle: `ACCEPT`, `LOG`, `DROP`...
Exemple :
```
# iptables -A INPUT -p icmp -j DROP
```
`-p --protocol` : Spécifier un protocole : `tcp`, `udp`, `icmp`, `all` (tous)
Exemple :
```
# iptables -A INPUT -p icmp -j DROP
```
`-s --source` : Spécifier une adresse source à matcher
Exemple :
```
# iptables -A INPUT -p tcp -s 192.168.42.42 -j ACCEPT
```
`-d --destination` : Spécifier une adresse destination
Exemple :
```
# iptables -A FORWARD -p tcp -d 10.1.0.1 -j ACCEPT
```
`-i --in-interface` : Spécifier une interface d'entrée
Exemple :
```
# iptables -A INPUT -p icmp -i eth0 -j DROP
```
`-o --out-interface` : Spécifier une interface de sortie
Exemple :
```
# iptables -A OUTPUT -p icmp -o eth0 -j DROP
```
`-f --fragment` : Paquet fragmenté
Exemple :
```
# iptables -A INPUT -p icmp -f -j DROP
```
`--sport --source-port` : Spécifier le port source ou une plage de ports, fonctionne aussi en udp, `-m multiport` permet de
spécifier plusieurs ports à matcher.
Exemples :
```
# iptables -A INPUT -p tcp --sport 80 -j ACCEPT
# iptables -A INPUT -p udp --sport 80 -j DROP
# iptables -A OUTPUT -p tcp -m multiport --sport 3128,21,1000 -j DROP
# iptables -A OUTPUT -p tcp --sport 1024:2042 -j ACCEPT
```
`--dport --destination-port` : Spécifier le port destination ou une plage de ports, fonctionne aussi en udp, `-m multiport`
permet de spécifier plusieurs ports a matcher.
Exemples :
```
# iptables -A INPUT -p tcp --dport 110 -j DROP
# iptables -A INPUT -p udp --dport 110 -j DROP
# iptables -A INPUT -p tcp -m multiport --dport 110,4242,119 -j DROP
# iptables -A INPUT -p tcp --sport 4925:4633 -j ACCEPT
```
`--tcp-flags` : Spécifier un flag tcp à matcher : `SYN ACK FIN RST URG PSH ALL NONE`
Exemple :
```
# iptables -A INPUT -p tcp --dport 42 --tcp-flags SYN,ACK -j ACCEPT
```
`--icmp-type` : Spécifier un type de paquet icmp à matcher
Exemple :
```
# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
```
`--mac-source` : Spécifier l'adresse MAC à matcher
Exemple :

```
# iptables -A INPUT --mac-source 42.42.AA.42.42.AA -j DROP
```
En cas d'erreur, il peut-être nécessaire de spécifier le module mac, exemple :
```
# iptables -A INPUT -m mac --mac-source 42.42.AA.42.42.AA -j DROP
```

Spécificités NAT :
`--to-destination` : Utilisé en target pour le *DNAT,* permet de spécifier l'adresse de destination de la translation, on peut également spécifier un port s'il est différent du port source.
Exemples :
```
# iptables -t nat -A PREROUTING -d 42.12.42.12 -p tcp --dport 110 -j DNAT --to-destination 192.168.1.2:6110
# iptables -t nat -A PREROUTING -d ! 42.12.42.12 -p tcp --dport 80 -j DNAT --to-destination 192.168.2.1:3128
```
`--to-source` : Utilisé pour en target pour le *SNAT,* permet de spécifier l'adresse source de la translation.

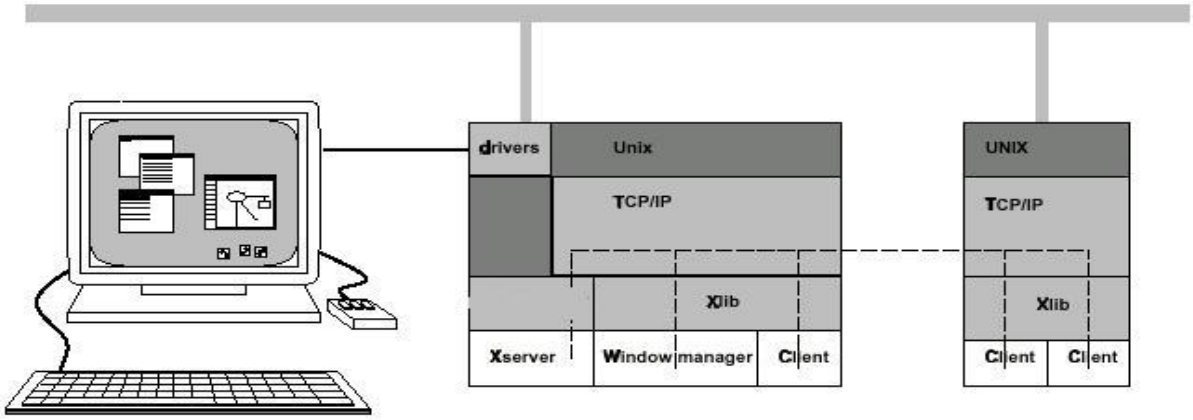
21.5 La sécurité X11

X11 est constitué des parties suivantes :

- un serveur : C'est un programme gérant du matériel (le display) et du pur logiciel : fenêtres, ressources, événements
- des clients : Des applications utilisant un serveur X.
  - émulation graphique d'un terminal : *xterm*
  - accessoires de bureau : *xclock*, *xbiff*, . . .
  - gestionnaire de fenêtres : *twm*, *fvwm*, *mwm*, *olwm*, *kde*, *gnome*, . .
    - C'est un client X, au même titre que les autres. Il en existe de nombreux.
    - Il permet de réaliser les choses suivantes : déplacer ou redimensionner une fenêtre, iconifier ou non une fenêtre, faire passer au premier ou au dernier plan une fenêtre, décorer les fenêtres, créer ou détruire les fenêtres, lancer ou terminer des applications
  - Logiciels de bureautique : *OpenOffice*...
  - Logiciels graphiques : *GIMP*, *Killustrator*…
- un protocole : Il fait communiquer les clients et le serveur.
  - X utilise un modèle client serveur
  - Sur IP il utilise les ports tcp 6000 à 6000+n n correspondant à votre numéro de display (en théorie 6000 à 6063)
- des bibliothèques : Elles réalisent le protocole et l'interface des clients

X11 est un protocole client/serveur où :

- le serveur X est le gestionnaire du terminal graphique c'est à dire qu'il gère l'écran, le clavier et la souris qui lui sont associés;
- le client X est l'application que l'on invoque depuis une machine quelconque du réseau pour effectuer des actions sur le terminal du serveur X : une émulation de terminal, une horloge, etc... Cependant le résultat de l'action d'un client ne se manifeste pas forcément sous forme graphique (par exemple lecture des caractéristiques du serveur ou encore lecture des évènements clavier) et là on comprend bien que certaines actions peuvent se faire à l'insu de l'utilisateur du terminal.



Dessin 5: Ptotocole X Window

X ne dispose pas d'un concept de privilège et les autorisations ne sont vérifiées qu'à la connexion. Un fois l'accès offert, vous pourrez :

- démarrer de nouveaux clients;
- enregistrer des événements;
- modifier des ressources attachées au serveur et supprimer des clients.

L'accès au serveur est contrôlé par les mécanisme de protection suivants :

- l'autorisation basée sur la reconnaissance d'une machine distante (autorisation par défaut). Ce mécanisme utilise la commande xhost pour gérer les autorisations :

xhost [+/-] [nom-de-machine]

Exemples:
`xhost + machine1` (Autorise la machine *machine1*)

`xhost - machine1` (Supprime les autorisations pour la *machine1*)

`xhost` (liste les autorisations préalablement définies)

`xhost +` (Autorise toutes les machines). Certains terminaux X sont fournis par défaut avec une autorisation de ce type c'est à dire complète insécurisé : **ceci doit être supprimé**.

Ce type de protection n'est réellement utilisable que pour des petites configurations réseau, avec peu d'utilisateurs puisque n'importe quel utilisateur d'une machine autorisée peut interagir avec le serveur. Il convient donc d'être très vigilant.

- des protocoles d'autorisation par utilisateur. Pour pouvoir interagir, un serveur et un client devront échanger une clé commune (appelée "cookie"); chaque utilisateur disposant de sa propre clé qu'il stockera dans un fichier. Il existe plusieurs mécanismes de clé mais nous ne parlerons ici que MIT-MAGIC-COOKIE-1. Ces mécanismes ne sont disponibles qu'à partir de la version 4 de X11.

C'est le fichier **.Xauthority** dans le répertoire "home" de l'utilisateur qui va maintenir cette clé. La commande **xauth** permettra d'en manipuler son contenu. Par exemple :

xauth add machine1:0 bb11acbb

générera une clé hexadécimale. Cette clé sera lue par le client X désirant accéder aux terminal de *machine1* et présentée au serveur X sur cette machine.

De son coté le serveur X doit être démarré de façon à tenir compte de ces clés. Deux cas se présentent :

- le serveur démarre grâce à la commande **xinit**. La syntaxe d'appel est la suivante : xinit $HOME/.xinitrc -- -auth $HOME/.Xauthority
- le serveur démarre en utilisant le protocole **XDM** (X Display Manager). Dans ce cas, XDM gère lui même la clé et son insertion dans *.Xauthority* sans intervention de l'utilisateur.

Il conviendra dans tous les cas de protéger *.Xauthority* contre la lecture par d'autres utilisateurs (mode **600**)

22 L'impression

Nous citerons brièvement 2 évolutions successives que sont :

- Le couple de commande historique lpr/lpd
- cups (pour Common Unix Printing System)

22.1 lpr/lpd

lpr est la commande d'impression depuis un client. **lpd** est le démon d'impression. En cas d'impression distante il doit tourner sur le client et le serveur d'impression.

lpd doit utiliser un fichier */etc/printcap* qui décrit les imprimantes disponibles.
lpd stocke les impressions en cours dans un dossier */var/spool/lpd* et gère les problèmes de contentions afin d'éviter qu'une imprimante ne reçoive plusieurs impressions simultanées.
L'installation de lpd est très simple ; il suffit de décrire les imprimantes dans ***printcap*** notamment les informations :

- lp : décrit le « device » sur lequel est connecté l'imprimante (par ex */dev/lp0*)
- rm : si l'imprimante n'est pas locale décrit le nom de la machine qui gère l'imprimante
- rp : décrit le nom de l' imprimante distante
- sd : nom du répertoire de spool
- lf : nom du fichier pour reporter les erreurs
- sh : suppression de la banière
- if : filtre d'entrée : le fichier sera filtré par un programme pour le convertir dans un format utilisable par l'imprimante

La commande **lpc** permet la gestion des queues d'impression. lpc sans argument rentre en mode interactif et permet des sous-commandes de type start, stop, enable, disable, abort, status….
lpc permet une utilisation directe du type : *lpc status mon_imprimante*

Dans l'impression sous Unix/Linux la difficulté majeure est de filtrer les documents à imprimer, afin qu'ils arrivent dans le bon langage pour l'imprimante. Dans le cas le plus courant il faut arriver à du Postscript en partant de texte, de PDF, d'images (jpeg, gif…), voire de formats plus exotiques. Il existe plusieurs ensembles de filtres qui sont appelés, comme nous l'avons vu, par la ligne *if* = du

fichier printcap.

22.2 CUPS

CUPS est la dernière génération des systèmes d'impressions sous Linux et devient le standard. C'est un énorme travail qui a été mené pour rendre l'impression simple sur tout système *UNIX*.
CUPS est très modulaire et doit permettre à tout vendeur d'imprimantes de rajouter de façon simple un pilote pour un modèle d'imprimante.
Il s'appuie sur un nouveau protocole réseau IPP (Internet Printing Protocol) surcouche de http, qui incluera à terme des possibilités d'encryption.
CUPS a généralisé la notion de *PPD* (*Postscript Printer Description*) à des imprimantes non postscript.
CUPS inclut lui-même des filtres pour nombre d'imprimantes.
CUPS peut collaborer avec les systèmes utilisant lpd.
Enfin CUPS fournit les commandes d'impressions lp et lpr avec des extensions très appréciables qui permettent de passer en ligne de commande de nombreuses options de configuration qu'il fallait auparavant inclure dans les fichiers de configuration de lpd (2 pages par feuille….etc)

CUPS s'administre avec une interface Web, en accédant en http sur le port 631 à la machine qui tourne le serveur cupsd.

Les fichiers de configuration se trouvent dans */etc/cups*, notamment le fichier ***cupsd.conf*** qui contient la configuration du service cupsd et ***printers.conf*** qui joue le rôle qu'avait printcap précédemment.

Il est possible de centraliser les impressions sur un seul serveur puis laisser les clients récupérer la liste des imprimantes depuis ce serveur. Pour obtenir cette fonctionnalité, on configure un serveur CUPS central sur lequel sont enregistrées toutes les imprimantes. Ce serveur CUPS peut diffuser l'ensemble des imprimantes au parc de clients.

Pour qu'un client récupère la liste des imprimantes depuis le serveur et puisse les utiliser, il doit contenir la directive suivante dans son fichier */etc/cups/client.conf* :

```
ServerName nom-du-serveur-cups
```

Si un fichier de configuration est modifié, il faut demander à CUPS de recharger ses fichiers de configuration : */etc/init.d/cups reload*

23 SAMBA

Samba est un ensemble de programmes qui permet d'offrir ou/et d'utiliser des ressources partagées via le jeu de commandes *SMB* (*Server Message Block*) issu de *NetBIOS* (*Network Basic Input/Output System*).
En clair, Samba permet à une machine *UNIX* de parler *Microsoft*, et ainsi de remplacer un serveur de fichiers et/ou d'imprimantes.

La configuration de Samba est contrôlée par un seul fichier **smb.conf** qui est dans */etc/samba/* pour un serveur standard. Un utilitaire de configuration WEB **swat** est disponible, mais il est prudent de faire une copie de sauvegarde du fichier de configuration avant de l'utiliser.

Exemple de fichier de configuration smb.conf

```
# Samba config file created using SWAT
# from lyopc128.in2p3.fr (134.158.139.178)
# Date: 2000/10/24 16:42:49
```

```
# Global parameters
[global]
workgroup = INFORMATIQUE
# si active directory, on utilise : realm
# realm = DOMAINE.ACTIVE.DIRECTORY
server string = %h server (Samba %v)
security = SHARE
# si active directory, on utilise : security = ADS
# password server = SERVEUR_AD1 SERVEUR_AD2
encrypt passwords = Yes
passwd program = /usr/bin/passwd %u
passwd chat = *Enter\snew\sUNIX\spassword:* %n\n \
*Retype\snew\sUNIX\spassword:* %n\n .
syslog = 0
max log size = 1000
socket options = IPTOS_LOWDELAY TCP_NODELAY SO_SNDBUF=4096\
SO_RCVBUF=4096
printcap name = /etc/samba/printcap
dns proxy = No
invalid users = root

# Homes parameters
[sharedisk]
comment = disk partage pour tous
path = /home/sharedisk
guest ok = Yes
hosts allow = 134.158.136.0/255.255.248.0

# Printers parameters
[printers]
comment = All Printers
path = /tmp
create mask = 0700
guest ok = Yes
printable = Yes
postscript = Yes
printing = lprng
browseable = No
```

Lorsque samba est démarré les démons **smbd** et **nmbd** tournent en machine.

Utilisation :

- Dans le cas ci-dessus, on pourra depuis un PC sous Windows connecter un lecteur réseau *serveursmb*\sharedisk et donc accéder depuis Windows à une zone disque du serveur Linux.

- De même depuis Windows on pourrait ajouter une imprimante réseau et voir les imprimantes partagées par smbd comme des imprimantes réseau valides.

- Depuis Linux on peut monter un partage Windows comme un point de montage classique :
  `smbmount –t smbfs //serveurWindows/partage /mnt/windows`
  ou
  `mount -t cifs -o username=forlinux //lyoforXX/test-linux /mnt`

- Faire une session pour transférer des fichiers :
  `mbclient -U stageadm -W formation //serveurWindows/partage`

smb: \>
(permet un équivalent de ftp mais vers un système Windows)

Samba peut faire beaucoup plus puisqu'il peut être contrôleur principal de domaine Windows.
Samba peut aussi être associé à cups pour qu'un serveur cups soit serveur d'impression Windows et il existe des « CUPS Driver for Windows » qui s'interfacent avec Samba..

24 ANNEXES

24.1 Commandes Unix

24.1.1 Informations système

uname - Identification du système.
-a : toutes les informations.
dmesg - Messages du noyau (et ceux du boot).
uptime - Durée et charge du système.
free - Occupation de la mémoire.
vmstat - Détails sur l'utilisation de la mémoire.
ipgcs - Utilisation des ressources IPC System V.
ipcrm - Suppression de ressources IPC System V.
ldconfig - Valider les bibliothèques dynamiques.
init - Changement de niveau de fonctionnement :
0 : arrêt.
1 : mono-utilisateur,
3 : multi-utilisateurs mode texte,
5 : multi-utilisateurs mode graphique,
6 : redémarrer.

24.1.2 Gestion des utilisateurs

useradd - Ajout d'un utilisateur :
useradd -m -p "" linus
crée un compte *linus*, avec répertoire personnel et mot de passe vide.
userdel - Suppression d'un compte utilisateur :
userdel -r linus
supprime le compte et le contenu de son répertoire.
passwd - Modification d'un mot de passe :
passwd linus

24.1.3 Partitions et systèmes de fichiers

fdisk - Édition de la table des partitions :
fdisk /dev/hda
mkswap - Création d'une zone de swap :
mkswap /dev/hda2
mkswap /boot/swap_file
swapon - Activation d'une zone de swap :
swapon /dev/hda2
-a active toutes les zones de swap de /etc/fstab.
swapoff - Désactivation d'une zone de swap :
swapoff /dev/hda2
mkfs - Création d'un système de fichiers :
mkfs.ext2 /dev/hda3

mkfs.ext3 /dev/hda4
mkfs.vfat /dev/hda5

fsck - Vérification d'un système de fichiers :
fsck.ext2 -p /dev/hda3
réparation automatique d'un système ext2/ext3,
fsck.vfat /dev/hda4
vérification d'une partition Windows.

mount - Insertion de partition dans le système :
mount -t vfat /dev/hda4 /mnt/dos/
monter une partition Windows,
mount -a
monter toutes les partitions de /etc/fstab,
mount 192.1.1.254:/home /home/users/
Montage d'un répertoire distant par NFS.
Options avec *-o* ou dans */etc/fstab* :
default : rw,suid,dev,exec,auto,nouser,async,
remount : changer les attributs d'un système monté,
rw : lecture-écriture,
ro : lecture seule,
noauto : ne pas monter automatiquement avec -a,
nodev : interdire les fichiers spéciaux,
noexec : pas de fichiers exécutables,
nosuid ; ignorer les bits Set-UID/GID,
sync : écritures synchrones,
user : peut être monté par un utilisateur.

Types de systèmes de fichiers courants :
ext2, ext3, msdos, vfat, proc, iso9660, udf, smb.

umount - Démontage d'un système de fichiers :
-a : démonte tous les systèmes dans /etc/mtab.
umount /dev/hda4
umount /mnt/dos
umount -a

df - Taux d'occupation des systèmes de fichiers montés.

24.1.4 Distribution/installation de logiciel

tar - Gestion d'archives :
-c : création d'archive,
-x : extraction d'archive,
-t : consultation d'archive,
-f : nom du fichier archive,
-v : mode volubile,
-z : (dé)compresser avec g(un)zip,
-j : (dé)compresser avec b(un)zip2.
tar -czf archive.tar.gz distrib/
crée une archive compressée du répertoire distrib/,
tar -tvf archive.tar
liste le contenu de l'archive,
tar -xjf archive.tar.bz2

extrait le contenu d'une archive compressée.
installation classique :

tar -xzf application-1.01.tar.gz
cd application-1.01
./configure
make && make install

rpm - Gestion des paquetages RedHat :

-h : affichage de la progression du travail.
rpm -ivh paquet.rpm
installation d'un paquetage,
rpm -Uvh paquet.rpm
mise à jour/installation d'un paquetage,
rpm -Fvh paquet.rpm
mise à jour d'un paquetage déjà installé,
rpm -e paquet
désinstallation d'un paquetage,
rpm -qa
liste de tous les paquetages installés,
rpm -qf /chemin/fichier
recherche du paquetage auquel appartient le fichier,
rpm -qip paquet.rpm
informations sur un paquetage,
rpm -qlp paquet.rpm
liste des fichiers contenus dans le paquetage.

apt - Gestion des paquetages Debian :

apt-get install paquetage
installation d'un paquetage et ressources éventuelles,
apt-get remove paquetage
suppression paquetage et dépendances éventuelles,
apt-get update
mise à jour de la base de données interne,
apt-get upgrade
mise à jour du système.
apt-get dist-upgrade
mise à jour « majeure » du système.

yum : Yellow dog Update

yum search xxx
Recherche la chaine xxx dans la description des paquetages
*yum list xxx**
Recherche la disponibilité des paquetages dont le nom commence par xxx
yum info xxx
affiche les informations sur le paquetage xxx
yum install xxx
installe le paquetage xxx et toutes les dépendances si nécessaire.
yum remove xxx
supprime le paquetage xxx et toutes les dépendances si nécessaire.
yum –enablerepo=dag install xxx
installe le paquetage xxx et toutes les dépendances si nécessaire en allant aussi chercher dans le repository dag.
yum –enablerepo= install xxx*

installe le paquetage xxx et toutes les dépendances si nécessaire en allant aussi chercher dans tous les repositories configurés dans YUM.

yum update

met à jour l'ensemble des paquetages du système

24.1.5 Gestion des processus

application &

lance l'application à l'arrière-plan,

***fg** 1*

ramène à l'avant-plan le job numéro 1,

(Ctrl-Z)

endort l'application à l'avant-plan,

bg

relance à l'arrière-plan un job endormi.

ps - État des processus :

ps -ef

ou

ps -aux

affichage long de tous les processus du système.

top - Affichage continu des processus du système.

-d délai de rafraîchissement.

renice - Changer la courtoisie d'un processus :

renice +5 12857

augmente la courtoisie du processus 12857 de 5 unités,

renice -5 -u root

diminue de 5 la courtoisie de tous les processus de root.

kill - Envoyer un signal à un processus :

kill -15 12857

-l (lettre l) : liste des signaux disponibles.

killall - Tuer tous les processus du même nom :

killall -9 boucle_fork

fuser - Liste des processus accédant à un fichier :

fuser -k -m /dev/hda5

tue tous les processus accédant à la partition indiquée.

24.1.6 Utilitaires réseau

ifconfig - Configuration des interfaces réseau :

ifconfig -a

affiche la configuration de toutes les interfaces réseau,

ifconfig eth0 192.1.1.50

configure la première interface ethernet.

route - Gestion de la table de routage du noyau :

route add -net 192.1.1.0 eth0

ajoute une route statique via l'interface eth0,

route add -net 172.1.1.0 gw 192.1.1.5

ajoute un réseau accessible par une passerelle,

route add default eth1

ajoute une route par défaut,
route del default
supprime la route par défaut.

socklist - Liste des sockets actives.

netstat - Statistiques réseau :
netstat -r
affiche la table de routage du noyau,
netstat -i
affiche l'état des différentes interfaces,
netstat -a
affiche l'état des sockets du système.

arp - Gestion de la table ARP du noyau :
-a affiche toutes les entrées dans le cache ARP,
arp -d hote
supprime les entrées concernant l'hôte indiqué.

ping - demande d'écho vers d'autres hôtes :
ping -c 1 -w 2 192.1.1.53
une seule requête et attend au plus 2 secondes,
ping -b 192.1.1.255
requête diffusée en broadcast à tous les hôtes du sous-réseau.

traceroute - Chemin pour joindre un hôte :
traceroute www.destination.com
-n ne pas traduire les adresses numériques en noms.

tcpdump - Examen du trafic réseau :
tcpdump -i eth0
affiche tout ce qui circule sur eth0,
tcpdump -i eth0 port telnet
affiche les message depuis / vers le port 23 (*telnet*).

telnet - Connexion TCP/IP :
telnet mail.isp.com pop-3
connexion sur port 110 (*Pop/3*) du serveur de courrier.

rsh - Exécution d'un shell distant.

ssh - Exécution sécurisée d'un shell distant.
ssh usera@192.168.1.54

ftp - Transferts de fichiers :
Commandes usuelles :
open ftp.serveur.org
cd /chemin/distant/
lcd /chemin/local/
get fichier
put fichier
prompt
*mget *.c*
*mput *.h*

wget - Rapatrier le contenu d'une URL :
wget http://www.site.com/repertoire/
-c reprendre un transfert déjà entamé,
-r charger récursivement les liens,
-l niveau maximal de récursion,
-k convertir les liens en pointeurs locaux.

24.1.7 Signaux fréquemment utilisés

0 : pseudo signal vérifiant la présence d'un processus,
1 (SIGHUP) : fin de connexion,
2 (SIGINT, Ctrl-C) : fin immédiate du programme,
3 (SIGQUIT, Ctrl-\) : fin immédiate avec fichier core,
9 (SIGKILL) : fin obligatoire et immédiate,
15 (SIGTERM) : fin normale.

Utilisés avec les commandes *kill* et *killall* :
Exemple : *killall -HUP xinetd*
kill -HUP 12345

24.1.8 Gestion des modules du noyau

lsmod - Liste des modules chargés.
modinfo - *Informations* sur un fichier module.
insmod - Insertion d'un module dans le noyau :
insmod module.o
rmmod - Suppression d'un modulé chargé :
rmmod module
depmod - Vérification des dépendances :
depmod -an
modprobe - Chargement gérant les dépendances :
modprobe module.o

24.2 Les commandes de d'éditeur Vi

Commande	Description
0	Aller en début de ligne
$	Aller en fin de ligne
k	Ligne précédente
j	Ligne suivante
h	Caractère précédent
l	Caractère suivant
b	Mot précédent
w	Mot suivant
<Ctrl-B>	Page précédente
<Ctrl-F>	Page suivante
nG	Aller à la ligne n. Exemple: 1G va sur la première ligne
G	Aller à la dernière ligne
x	Supprime le caractère sous le curseur
dd	Supprime la ligne courante et la copie dans le presse-papiers
nd	Idem avec n lignes
J	Fusionne la ligne courante et la suivante
yy	Copie la ligne courante dans le presse-papiers
ny	Idem avec n lignes
P	Colle le presse-papiers avant la position courante
p	Colle le presse-papiers après la position courante
v	Commence une sélection en mode caractères
V	Commence une sélection en mode lignes
<Ctrl-V>	Commence une sélection en mode "rectangulaire"
d	Supprime la sélection et la copie dans le presse-papiers
y	Copie la sélection dans le presse-papiers
c	Supprime la sélection et passe en mode insertion
i	Passe en mode insertion avant la position courante
a	Passe en mode insertion après la position courante
o	Passe en mode insertion sur une nouvelle ligne sous la ligne courante
<ESC>	Quitte le mode insertion
u	Annule la dernière commande
r	Remplace le caractère sous le curseur par le prochain caractère tapé
~	Convertit le caractère sous le curseur en majuscule si c'est une minuscule et vice-versa
/texte	Recherche en avant du texte indiqué
?texte	Recherche en arrière du texte indiqué
n	Recherche l'occurrence suivante
N	Recherche l'occurrence précédente
:%s/chercher/remplacer	Recherche avec remplacement dans tout le fichier
:w	Sauvegarde le fichier courant
:wfichier	Ecrit le document dans le fichier indiqué
:rfichier	Inclut le fichier indiqué à partir de la position courante
:q!	Quitter en annulant les modifications
ZZ (ou :wq)	Quitter en enregistrant les modifications