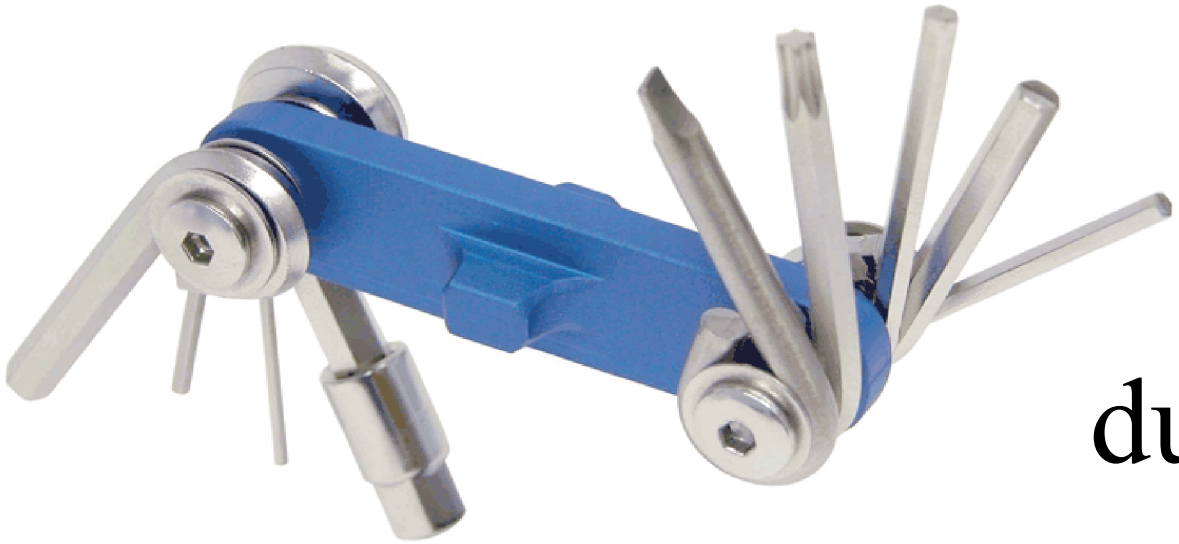
Support de cours



Les bases de l'administration du système *Linux*

Pré-requis et déroulement

Ce support de cours a pour objet la formation aux bases de l'administration du système Linux. La pratique du système Unix ou Linux d'un point de vue utilisateur n'y est pas abordée.

En particulier, on attend de chaque stagiaire qu'il :

- maîtrise l'utilisation d'Unix (commandes *shell* de base) ;
- sache modifier des fichiers au format texte à l'aide de l'un des éditeurs traditionnellement disponibles sous Unix (*vi*, *emacs*, *pico*, etc.).

La durée théorique de ce module de formation est d'une journée et demie.

Il s'agit d'une formation interactive, il est donc tout à fait indiqué d'interrompre le formateur pour lui poser des questions, lui faire préciser certains points, demander l'étude d'un cas particulier, etc.

Responsabilités de l'administrateur

Trois grandes familles de tâches incombent à l'administrateur Unix : gérer le système, les services et la sécurité.

Surveiller et assurer la bonne marche du système au quotidien :

- surveiller les ressources (disque, mémoire, CPU, etc.) ;
- planifier l'ajout de ressources.

Administrer les services déployés :

- gérer les utilisateurs ;
- installer et configurer les applications ;
- planifier les migrations.

Prévoir et gérer les incidents et les intrusions (tâches transversales) :

- installer les correctifs de sécurité ;
- « durcir » le système et les applications ;
- mettre en oeuvre un plan de sauvegarde ;
- superviser le système et les applications.

L'utilisateur « root »

Pour réaliser les tâches d'administration, on utilise le compte traditionnellement appelé « root » (à ne pas confondre avec la racine de l'arborescence qui porte le même nom).

De ce point de vue, le système de gestion des droits à la mode Unix est plutôt primitif, un utilisateur spécial dispose des droits lui permettant d'accéder à l'intégralité du système. Tous les autres utilisateurs sont soumis au système de gestion des permissions d'accès.

Il est donc assez difficile, sous Unix, de déléguer l'administration d'un sous-système à des utilisateurs.

Arborescence

La disposition des fichiers dans l'arborescence des systèmes Unix n'est, en général, pas d'une compréhension évidente de prime abord.

Il est cependant indispensable de connaître les règles qui président à la distribution des fichiers et répertoires dans le système des fichiers afin de :

- déterminer un plan de sauvegarde ;
- gérer la sécurité ;
- agir efficacement lors de la résolution des problèmes ;
- installer des logiciels non disponibles sous formes de paquetage ;
- etc.

Un document, le FHS (*Filesystem Hierarchy Standard* disponible sur cette page :`http://www.pathname.com/fhs/`), a pour ambition de proposer la normalisation de l'organisation de système de fichiers pour les systèmes Unix.

La plupart des distributions Linux s'y conforme même s'il reste beaucoup de différences dans les détails d'implémentation.

La racine du système de fichiers

La racine de l'arborescence Linux contient ces répertoires :

`bin`	programmes utilisateur essentiels (nécessaires au démarrage du système)
`boot`	fichiers nécessaires au chargement de Linux (*bootloader*, *initrd*, noyau)
`dev`	fichiers spéciaux offrant l'accès aux périphériques
`etc`	configuration du système et des services
`home`	répertoires principaux des utilisateurs
`initrd`	répertoire utilisé pour construire l'image disque en RAM du noyau
`lib`	librairies partagées essentielles (nécessaires au démarrage du système)
`mnt`	contient des points de montage temporaires (`cdrom`, `floppy`, etc.)
`opt`	applications tierces
`proc` et `sys`	systèmes de fichiers virtuels permettant d'accéder aux structures internes du noyau (`sys` est nouveau depuis la version 2.6 de Linux)
`root`	répertoire principal de l'utilisateur *root*
`sbin`	exécutables système essentiels (nécessaires au démarrage du système)
`tmp`	répertoire pour le stockage de fichiers temporaires
`usr`	arborescence contenant la plupart des fichiers des applications
`var`	données vivantes du système et des applications

`/etc` et `/lib`

`/etc`	fichiers et répertoires de configuration du système et des applications
`/etc/X11`	configuration du système X11
`/etc/rc*`	répertoires et scripts utilisés lors du démarrage du système
`/etc/pam.d`	configuration, par service, des *Pluggable Authentication Modules*

Lorsqu'une application utilise plusieurs fichiers de configuration, un sous-répertoire de `/etc` lui est généralement dédié. Par exemple :

`/etc/httpd`	fichiers de configuration du serveur Web *Apache*
`/etc/mail`	fichiers de configuration du système de messagerie *Sendmail*
`/etc/ssh`	fichiers de configuration et clés asymétriques d'*OpenSSH*

`/lib`	librairies partagées essentielles
`/lib/modules`	modules du noyau
`/lib/security`	librairies PAM (*Pluggable Authentication Modules*)
`/lib/iptables`	greffons *iptables*
`/lib/kbd`	codages clavier et polices de la console

/usr

`/usr/bin`	la plupart des programmes utilisateur
`/usr/games`	jeux et autres facéties
`/usr/include`	en-têtes standards pour le développement
`/usr/lib`	librairies (`.so` pour *Shared Objects*) et autres ressources partagées
`/usr/libexec`	binaires exécutables appelés par d'autres programmes
`/usr/local`	programmes installés « à la main » (*i.e.* sans paquetage)
`/usr/sbin`	exécutables système
`/usr/share`	ressources partagées indépendantes de la plate-forme (pages de manuel, documentation, fichiers de données, etc.)
`/usr/src`	sources du système (noyau, paquetages, etc.)

Fichiers de la distribution *X Window System, Version 11 Release 6* :

`/usr/X11R6/bin`	exécutables
`/usr/X11R6/include`	en-têtes pour le développement en C
`/usr/X11R6/lib`	librairies et autres ressources partagées
`/usr/X11R6/man`	pages de manuel
`/usr/X11R6/share`	ressources partagées indépendantes de la plate-forme

/var

/var/account	journaux de comptabilité des ressources utilisées par les processus
/var/cache	données cachées par les applications
/var/lib	données vivantes et persistantes des applications
/var/lib/rpm	base de données des paquetages installés
/var/lock	fichiers verrous
/var/lock/subsys	contient un fichier pour chaque service actif
/var/log	journaux du système et des applications
/var/run	données vivantes, non-persistantes des applications
/var/spool	données en attente d'un traitement
/var/spool/mail	boîtes aux lettres des utilisateurs
/var/spool/cron	tâches planifiées par les utilisateurs
/var/spool/lpd	files pour le système d'impression
/var/spool/mqueue	messages en attente de traitement par le MTA
/var/tmp	fichiers temporaires préservés entre deux démarrages
/var/www	racine du serveur Web

Paquetages

Un paquetage contient :

- les fichiers (exécutables, librairies, fichiers de configuration, ressources, etc.) nécessaires pour exécuter un programme ;
- un ensemble de scripts qui configurent le programme automatiquement après son installation ;
- les informations sur le propriétaire et permissions d'accès de chaque fichier ;
- des informations optionnelles sur les paquetages dépendants et les services fournis ;
- une description du paquetage.

L'ensemble des informations de tous les paquetages installés sont stockés dans une base de données gérée par l'utilitaire `rpm` (*Redhat Package Management*) pour les distributions basées sur RedHat et les outils `dpkg` et `apt` pour la distribution Debian.

rpm

Le nom d'un fichier de paquetage est constitué du nom du paquetage, du numéro de version du logiciel et du numéro de version du paquetage séparés par des caractères « - ». Par exemple : `openssl-0.9.7a-35.rpm`

Exemples d'utilisation de *rpm* :

* installation d'un paquetage : `rpm -i fichier.rpm`
* mise-à-jour d'un paquetage : `rpm -U fichier.rpm`
* suppression d'un paquetage : `rpm -e paquetage`
* vérification d'un paquetage : `rpm -V paquetage`
* interrogation sur les paquetages installés :
  - liste des fichiers d'un paquetage : `rpm -ql paquetage`
  - paquetage contenant un fichier : `rpm -qf /chemin/vers/fichier`
  - informations sur un paquetage : `rpm -qi paquetage`
  - liste triée de tous les paquetage installés : `rpm -qa |sort`
* interrogation sur un fichier de paquetage :
  - liste des fichiers : `rpm -qpl fichier.rpm`
  - informations : `rpm -qpi fichier.rpm`

rpm – Gestion des signatures GPG

rpm supporte désormais la signature des paquetages. Cette technique permet de garantir qu'un paquet n'a pas subi de modification et que la personne qui l'a construit est bien celle qu'elle prétend être.

Il faut pour cela importer les clés publiques GPG des personnes ou organisations qui réalisent les paquetages devant être installés avec l'option `--import` de *rpm*. L'option `--checksig` permet de vérifier la signature d'un paquetage avant de l'installer.

La clé publique de la distribution RedHat est disponible sur les CD-ROM, dans le fichier `RPM-GPG-KEY` (ou `RPM-GPG-KEY-fedora` en ce qui concerne la Fedora).

Par exemple :

```
# rpm --checksig webmin-1.150-1.noarch.rpm
webmin-1.150-1.noarch.rpm: md5 (GPG) PAS OK (CLES MANQUANTES: GPG#11f63c51)
# wget http://www.webmin.com/jcameron-key.asc
.../...
# rpm --import jcameron-key.asc
# rpm --checksig webmin-1.150-1.noarch.rpm
webmin-1.150-1.noarch.rpm: md5 gpg OK
```

rpm – Gestion des signatures GPG

Pour lister les clés installées :

```
# rpm -qa gpg-pubkey*
gpg-pubkey-11f63c51-3c7dc11d
```

Pour avoir plus d'informations sur une clé particulière :

```
# rpm -qi gpg-pubkey-11f63c51-3c7dc11d
Name        : gpg-pubkey                   Relocations: (not relocatable)
Version     : 4f2a6fd2                         Vendor: (none)
Release     : 3f9d9d3b                      Build Date: ven 23 jui...
Install Date: ven 23 jui...                 Build Host: localhost
Group       : Public Keys                   Source RPM: (none)
Size        : 0                                License: pubkey
Signature   : (none)
Summary     : gpg(Fedora Project <fedora@redhat.com>)
Description :
-----BEGIN PGP PUBLIC KEY BLOCK-----
.../...
-----END PGP PUBLIC KEY BLOCK-----
```

Pour supprimer une clé :

```
# rpm -e gpg-pubkey-11f63c51-3c7dc11d
```

dpkg – Gestion des paquetages Debian

Le nom d'un fichier de paquetage est constitué :

* du nom du paquetage suivi ;
* du numéro de version du logiciel et du numéro de version du paquetage ;
* de l'architecture cible ;
* de l'extension `.deb`.

Par exemple : `openssl_0.9.7e-3sarge1_i386.deb`

Exemples d'utilisation de *dpkg* :

* installation d'un paquetage : `dpkg -i fichier.deb`
* suppression d'un paquetage : `dpkg -P paquetage`
* liste des paquetages installés : `dpkg -l`
* afficher des informations sur un paquetage : `dpkg -I fichier.deb`
* afficher la liste des fichiers installés par un paquetage : `dpkg -L paquetage`

apt – Dépôts de logiciels

Un dépôt Debian est un ensemble organisé et indexé de paquetages Debian et que l'infrastructure *apt* (*Advanced Packaging Tool*) de gestion des paquetages Debian sait utiliser pour installer des applications.

Un dépôt Debian peut être stocké sur différents supports :

* CD-ROM ;
* DVD-ROM ;
* un répertoire sur un système de fichiers ;
* un emplacement sur le réseau accessible en HTTP ou FTP ;
* etc.

Il existe plusieurs types de dépôts Debian :

* les dépôts officiels de la distribution ;
* le dépôt officiel contenant les correctifs de sécurité ;
* le dépôt « volatile » ;
* des dépôts Debian non officiels.

Afin qu'`apt` puisse utiliser un dépôt il faut :

1) inscrire ce dépôt dans le fichier `/etc/apt/sources.list` ;
2) créer un catalogue des paquetages disponibles dans ce dépôt.

apt – Dépôts de logiciels

Un dépôt Debian est un ensemble organisé et indexé de paquetages Debian et que l'infrastructure *apt* (*Advanced Packaging Tool*) de gestion des paquetages Debian sait utiliser pour installer des applications.

Un dépôt Debian peut être stocké sur différents supports :

* CD-ROM ;
* DVD-ROM ;
* un répertoire sur un système de fichiers ;
* un emplacement sur le réseau accessible en HTTP ou FTP ;
* etc.

Il existe plusieurs types de dépôts Debian :

* les dépôts officiels de la distribution ;
* le dépôt officiel contenant les correctifs de sécurité ;
* le dépôt « volatile » ;
* des dépôts Debian non officiels.

Afin qu'`apt` puisse utiliser un dépôt il faut :

1) inscrire ce dépôt dans le fichier `/etc/apt/sources.list` ;
2) créer un catalogue des paquetages disponibles dans ce dépôt.

Utilisateurs Unix

Les attributs qui caractérisent un utilisateur Unix sont :

- un nom de connexion (*login*) ;
- un mot de passe ;
- un identifiant numérique unique (UID) ;
- un groupe primaire (GID) ;
- un commentaire (appelé *gecos*);
- le répertoire principal de l'utilisateur (*home directory*) ;
- un interpréteur de commandes (*shell*) par défaut.

Un utilisateur est identifié par le système par son UID. L'utilisateur root a pour UID 0. *C'est cette caractéristique qui lui confère un accès complet au système.*

L'ensemble de ces éléments est stocké dans le fichier `/etc/passwd` au format texte. Les champs sont séparés par le caractère « : ». Par exemple :

```
jo:x:500:500:Jo Dalton:/home/jo:/bin/bash
```

Groupes Unix

Les attributs qui caractérisent un groupe Unix sont :

- un nom ;
- un mot de passe (jamais utilisé) ;
- un identifiant numérique unique (GID) ;
- une liste d'utilisateurs membres.

La liste des utilisateurs peut être vide ou contenir un plusieurs nom d'utilisateurs séparés par un caractère « , ».

L'ensemble de ces éléments est stocké dans le fichier `/etc/group` au format texte. Les champs sont séparés par le caractère « : ». Par exemple :

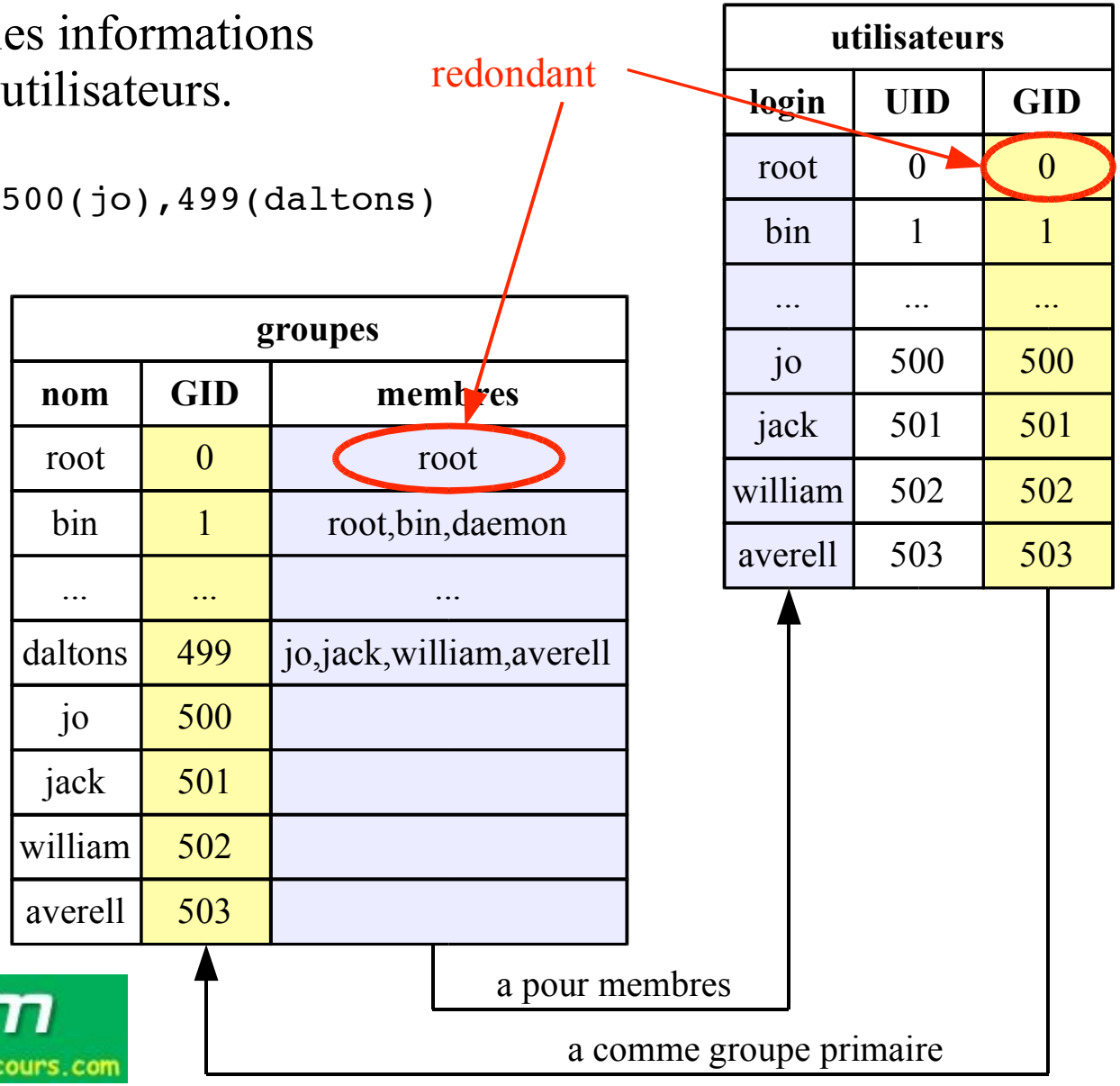
```
daltons:x:100:jo,jack,william,averell
```

Relations utilisateurs/groupes

La commande `id` permet de lister les informations (UID, GID, groupes) relatives aux utilisateurs.

```
$ id jo
uid=500(jo) gid=500(jo) groupes=500(jo),499(daltons)
```

(Voir également la commande `groups`.)



Conventions

En général, la liste des utilisateurs est segmentée ainsi :

- l'utilisateur dont l'UID est 0 est le root ;
- les utilisateurs dont l'UID est inférieur à une certaine valeur sont des utilisateurs systèmes ;
- les utilisateurs dont l'UID est compris entre cette valeur et 65533 sont des utilisateurs réels, cette plage peut quelquefois être elle-même subdivisée (par exemple lors de l'utilisation de NIS ou Winbind) ;
- l'utilisateur dont l'UID est 65534 est « nobody » ou « nfsnobody » (pour les systèmes qui supportent les UID sur 32 bits, afin de conserver une compatibilité avec les anciens serveurs NFS).

Souvent, à chaque utilisateur correspond un groupe :

- dont le GID est identique à l'UID de l'utilisateur ;
- dont le nom est le même que celui de l'utilisateur ;
- qui est le groupe primaire de l'utilisateur.

Le support des utilisateurs codés sur 32 bits commence à arriver dans les distributions récentes afin de porter le nombre d'utilisateurs possibles de 65535 à plus de 4 millions.

Mots de passe cachés

À l'origine d'Unix, les mots de passe étaient stockés dans le fichier `/etc/passwd` (deuxième champs) cryptés avec l'algorithme traditionnel *crypt*.

Comme ce fichier doit être accessible en lecture pour tout les utilisateurs, le stockage des mots de passe à été déplacé vers le fichier `/etc/shadow` dont l'accès en lecture est restreint (technique dite des « *shadow passwords* »). De plus, l'algorithme d'encodage MD5 est de plus en plus utilisé.

Par ailleurs des champs supplémentaires ont été ajoutés à ce fichier afin de permettre la gestion de l'expiration des mots de passe.

```
jo:c8NuTn3ScFJfA:12460:0:99999:7:::
```

Le champs du fichier `/etc/passwd` destiné à contenir le mot de passe, est généralement positionné à la valeur « x ».

Gestion des utilisateurs et groupes

Créer un utilisateur consiste à réaliser plusieurs étapes :

- créer l'utilisateur et éventuellement son groupe primaire dans les fichiers `passwd`, `shadow` et `group` ;
- créer le répertoire principal de cet utilisateur (généralement dans `/home`) et lui affecter les bons droits d'accès ;
- associer des quotas à l'utilisateur si nécessaire ;
- copier un certain nombre de fichiers standards depuis le répertoire `/etc/skel` vers le répertoire principal de l'utilisateur et leur affecter les bonnes permissions.

Ces opérations peuvent être réalisées :

- « à la main » (avec *vi*, `mkdir`, `cp`, `chown`, `chfn`, `chsh`, etc.), peu recommandé ;
- en ligne de commande avec principalement l'outil `adduser` ;
- à l'aide d'outils graphiques tels que Webmin ou le gestionnaires d'utilisateurs de RedHat.

Le principe reste le même pour la modification et la suppression des utilisateurs ou la gestion des groupes d'utilisateurs.

Le planificateur de tâches *cron*

Le planificateur de tâche *cron* permet à chaque utilisateur de configurer l'exécution périodique de commandes *via* l'utilisation de la commande `crontab`.

L'utilisateur *root* dispose également de cette possibilité. Cependant, pour installer l'exécution périodique de tâches relatives à la gestion du système, on préférera l'utilisation des répertoires suivants :

- `/etc/cron.hourly` pour une exécution toutes les heures ;
- `/etc/cron.daily` pour une exécution tous les jours ;
- `/etc/cron.weekly` pour une exécution hebdomadaire ;
- `/etc/cron.monthly` pour une exécution mensuelle ;
- et `/etc/cron.d` qui permet d'affiner plus précisément la planification.

L'utilisation des quatre premiers répertoires est simple : il suffit d'y copier un fichier exécutable (ou de créer un lien) afin d'en activer l'exécution périodique.

Le planificateur de tâches *cron*

L'utilisation du répertoire `/etc/cron.d` est plus particulière. Les fichiers contenus dans ce répertoires sont des fichiers texte qui adoptent un format quasi-identique à celui des fichiers manipulés par la commande *crontab*.

La seule information ajoutée est l'utilisateur qui sera utilisé pour exécuter la commande. Par exemple :

```
MAILTO=root
#mn hr day month dow user     command
0   4  *   *     *   postgres /usr/lib/postgresql/bin/do.maintenance -a
```

Le fichier `/etc/crontab` contrôle les jours et heures d'exécution des commandes présentes dans les répertoires `/etc/cron.*` (excepté `/etc/cron.d`).

Les planifications créées par les utilisateurs sont stockées à raison d'un fichier pour chaque utilisateur dans le répertoire `/var/spool/cron`.

Collecte des journaux de messages (*logs*)

Le mécanisme Unix traditionnel pour la collecte des messages générés par le système ou les applications est *Syslog*.

`syslogd` et `klogd` sont les programmes (démons) utilisés pour cela. `syslogd` se charge de la collecte et répartition des messages, `klogd` n'a pour tâche que de passer les messages du noyau à `syslogd`.

Il est possible de rediriger ces messages selon leur origine et/ou leur niveau vers des fichiers, la console, etc. voire vers le *Syslog* d'une autre machine.

Le fichier de configuration de `syslogd` est `/etc/syslog.conf`.

Par défaut, la quasi totalité des messages sont dirigés vers un ensemble de fichier situés dans le répertoire `/var/log`.

syslog.conf

Une ligne du fichier `/etc/syslog.conf` est de la forme :

```
facility.priority[,facility.priority,...]  action
```

Les sources des messages sont appelées « facilités ». En voici la liste :

`auth`	message concernant la sécurité ou l'authentification
`authpriv`	comme `auth` mais susceptible de contenir des informations privées
`cron`	messages générés par les planificateurs de tâches *cron* ou *at*
`daemon`	un des démons du système sans classification particulière
`kern`	messages du noyau
`lpr`	messages du sous-système d'impression
`mail`	messages du sous-système de messagerie
`mark`	marqueurs de temps générés par *Syslog*
`news`	messages du sous-système de gestion USENET
`syslog`	messages internes de *Syslog*
`user`	messages utilisateur génériques
`uucp`	messages du sous-système UUCP
`local0` à `local7`	facilités utilisées à la discrétion de l'administrateur local

`syslog.conf`

Les niveaux de priorités (par ordre de gravité) sont :

`debug`	messages destinés au *débuggage*
`info`	informations
`notice`	normal mais significatif
`warning`	avertissement
`err`	conditions d'erreur
`crit`	conditions critiques
`alert`	action à gérer immédiatement
`emerg`	système inutilisable

Une priorité représente également toutes les priorités qui lui sont supérieures excepté si elle est précédée du signe « = » (dans ce cas, elle ne représente qu'elle-même).

La priorité « `none` » indique de ne pas tenir compte du message pour la facilité concernée tandis que « `*` » représente tous les niveaux.

Le caractère « `!` » utilisé devant la priorité indique de ne pas en tenir compte.

Exemple commenté de `syslog.conf`

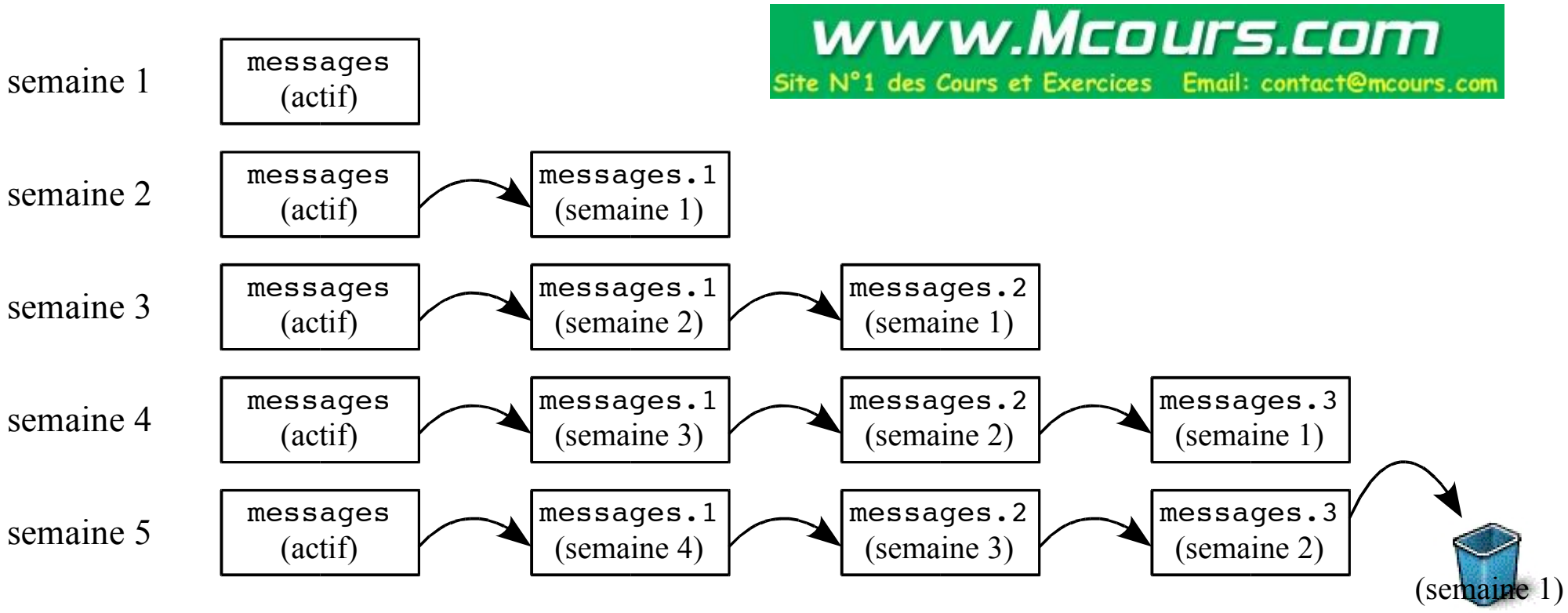
```
# Tous les messages du noyau sont envoyés sur la console.
kern.*                                          /dev/console

# Les messages dont le niveau n'est pas debug et qui ne concernent pas
# les facilités mail, authpriv et cron sont collectés dans un fichier.
*.info;mail.none;authpriv.none;cron.none  /var/log/messages

# Tous les messages ayant le niveau debug uniquement (excepté ceux avec
# la facilité authpriv) sont envoyés dans un fichier spécifique.
*.=debug;authpriv.none                          /var/log/debug

# Informations sensibles relatives à la sécurité.
authpriv.*                                      /var/log/secure

# Messages du sous-système de messagerie.
mail.info                                       /var/log/maillog

# Messages du planificateur de tâches.
cron.*                                          /var/log/cron

# Tous les utilisateurs connectés reçoivent les messages d'urgence.
*.emerg                                         *

# Tous les messages sont envoyés vers le Syslog d'une autre machine.
*.*                                             @supervision
```

Archive des journaux – `logrotate`

L'outil `logrotate` est utilisé pour effectuer des rotations automatiques et régulières (*via* le planificateur *cron*) des journaux produits par *Syslog* ou d'autres applications.

Le fichier de configuration est `/etc/logrotate.conf`. Chaque sous-système peut rajouter son propre fichier de configuration dans le répertoire `/etc/logrotate.d`.

Voici une illustration du travail que réalise `logrotate` sur une base de 3 rotations :



Démarrage du système

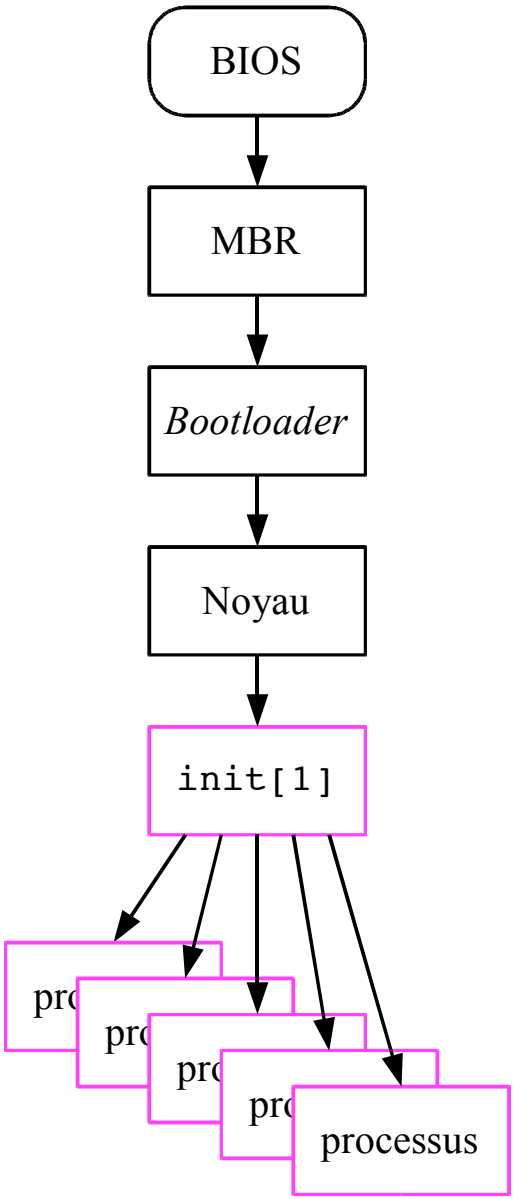
Juste après avoir démarré, initialisé le matériel et monté la partition racine en lecture seule, le noyau démarre le processus `/sbin/init`.

`init` est l'ancêtre de tous les autres processus du système (cela est visible avec la commande `pstree`).

Les programmes exécutés par `init` au démarrage sont décrits dans le fichier `/etc/inittab`. Ils sont fonctions du *runlevel* par défaut.

Le fichier `/etc/inittab` décrit également les actions à entreprendre lorsque certains évènements surgissent :

- perte de tension sur l'onduleur ;
- retour de tension sur l'onduleur ;
- séquence de touches « Ctrl-Alt-Del ».



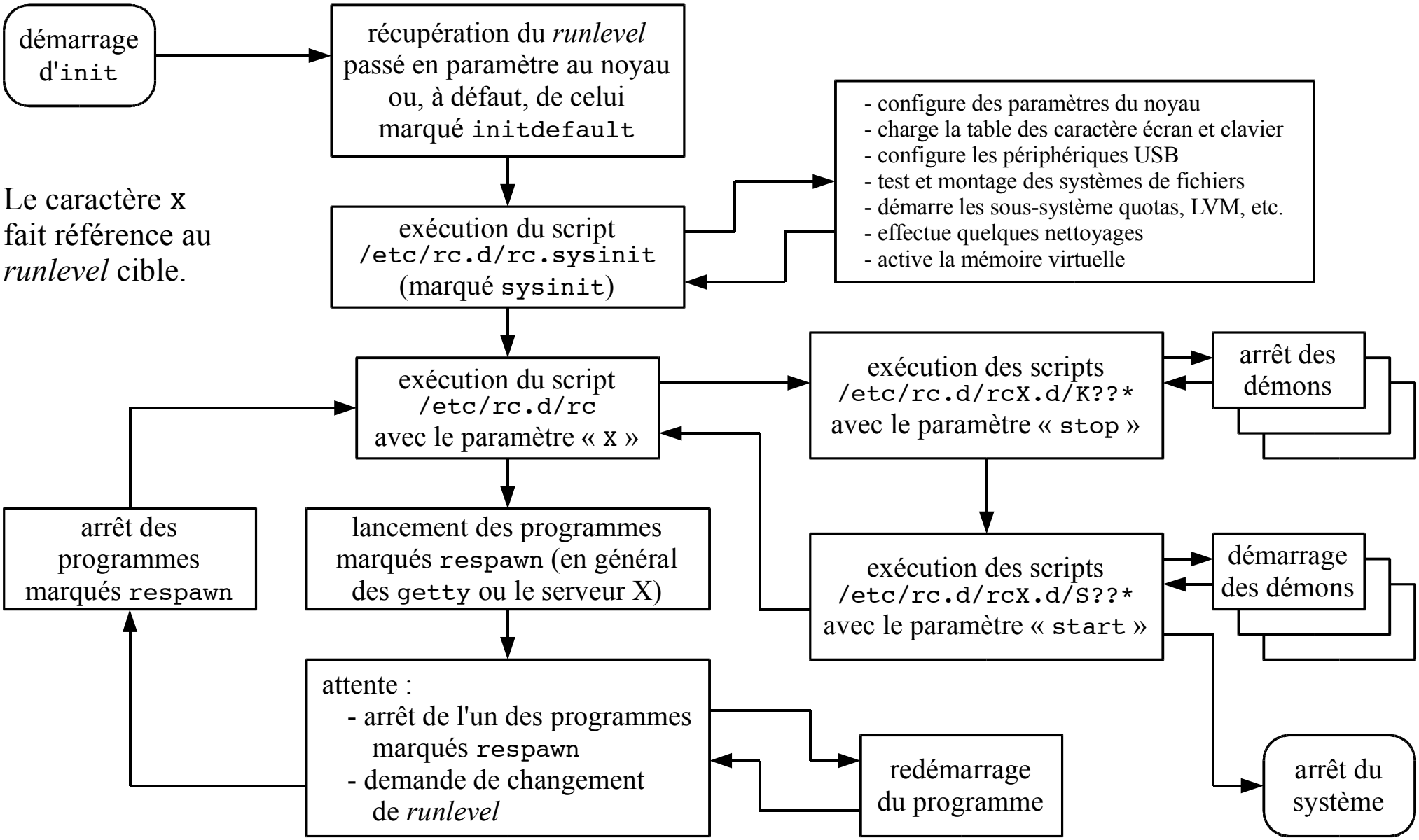
Runlevels

Un *runlevel* décrit un état du système. Il existe sept *runlevels* distincts.
Caque *runlevel* correspond à une utilisation du système :

0 arrêt du système
1 mode de maintenance (*single user mode*, « s » ou « S » peuvent être utilisés)
2 mode normal sans NFS
3 mode normal
4 inutilisé
5 mode normal avec X Window (défaut si un environnement Gnome ou KDE est installé)
6 redémarrage du système

Les *runlevels* 0, 1 et 6 sont toujours utilisés tels que décrits ci-dessus.
L'usage des autres *runlevels* peut varier en fonction de la distribution de Linux.

Démarrage – le processus complet



Changer de *runlevel*

Un *runlevel* précis peut être atteint :

- en le passant comme paramètre au noyau dans la configuration du chargeur de démarrage ou en modifiant de manière temporaire les paramètres de démarrage du noyau ;
- en utilisant les commandes `init` ou `telinit` lorsque le système est démarré, il est ainsi possible de passer d'un *runlevel* à l'autre sans arrêter le système ;
- en le configurant comme *runlevel* par défaut (pour le prochain démarrage du système) dans le fichier `/etc/inittab`, à la ligne `initdefault` ;
- en utilisant la commandes `shutdown` qui envoie un signal à `init`.

Avec le chargeur de démarrage GRUB (*GRand Unified Bootloader*), les paramètres de démarrage par défaut du noyau sont situés dans le fichier `/boot/grub/grub.conf` (lignes `kernel`).

Une interface rudimentaire permettant de modifier de manière temporaire les paramètres du noyau est disponible lors de la phase de démarrage de GRUB en appuyant sur la touche « e ».

Activer/désactiver des services au démarrage

Les services activés dans un *runlevel* sont ceux pour lesquels il existe un lien `/etc/rc.d/rcX.d/SNNservice` vers `/etc/rc.d/init.d/service` où :

- `X` est le *runlevel* cible ;
- `NN` est un numéro qui sert à déterminer l'ordre de démarrage des services ;
- `service` est le nom d'un script dans le répertoire `/etc/rc.d/init.d` qui, utilisé avec les paramètres « start » ou « stop » sait comment arrêter ou démarrer le service.

Le principe est le même pour l'arrêt des services avec des liens dont le nom est : `/etc/rc.d/rcX.d/KNNservice`.

Ces liens peuvent être gérés:

- « à la main » (commandes `ln -s` et `rm`) ;
- avec l'utilitaire en ligne de commande `chkconfig` (ou `update-rc.d` pour Debian) ;
- avec le programme console `ntsysv` ;
- avec une interface graphique telle que Webmin ou l'outil de gestion des services de RedHat.

Arrêter/démarrer des services

La commande `service` est utilisée pour arrêter ou démarrer les services.
Elle s'utilise ainsi :

```
# service nom action
```

Où `nom` est le nom d'un service (tel que présent dans le répertoire `/etc/init.d`) et `action` le type d'action à entreprendre :

- `stop` arrête le service ;
- `start` démarre le service ;
- `reload` demande au service de relire sa configuration ;
- `status` affiche l'état du service.

On peut également utiliser cette syntaxe :

```
# /etc/init.d/nom action
```

Les systèmes de fichiers

Un système de fichiers est une collection organisées de répertoires et fichiers selon un format donné. Linux reconnaît en lecture et en écriture plusieurs dizaines de format de systèmes de fichiers.

Selon le type de système de fichiers, le support servant à la persistance des données peu prendre l'une des formes suivantes (les exemples donnés entre parenthèses ne sont pas exhaustifs) :

- partition sur un périphérique géré par un contrôleur de disque local (disque, contrôleur RAID matériel) ;
- périphérique amovible (disquette, CD-ROM) ;
- ressource mise à disposition sur le réseau (tel que serveurs NFS ou SMB) ;
- mémoire vive (utilisée par le système de fichiers *tmpfs*) ;
- espace virtuel (ce qui est le cas de systèmes de fichiers tels que *proc*, *sysfs*, *usbdevfs* ou *devpts*) ;
- périphérique de type bloc émulé (par exemple fournis par *ramdisk*, *loop*, NBD, LVM, RAID logiciel).

Créer un système de fichiers

Le système de fichiers le plus utilisé est *ext3*. Pour créer un nouveau système de fichiers et l'attacher de manière permanente à l'arborescence, il est nécessaire de :

1) configurer un périphérique de type bloc ;
2) créer le système de fichiers avec la commande `mkfs` ;
3) choisir un point de montage dans l'arborescence ;
4) attacher le système de fichiers au point de montage avec la commande `mount` ;
5) ajouter une ligne au fichier `/etc/fstab` afin que le système de fichiers soit attaché au point de montage à chaque démarrage du système.

Les actions à mener pour réaliser la première étape dépendent du type de support utilisé pour recevoir le système de fichiers. La plupart du temps il s'agit d'une partition sur un disque local qui sera créée avec la commande `fdisk`.

Les partitions Linux ont un identifiant système 0x83. C'est l'identifiant qui est positionné par défaut lorsqu'une partition est créée avec la commande `fdisk` de Linux.

Créer un système de fichiers

Exemple de création d'une partition (`hda9`) et d'un système de fichiers :

```
# fdisk /dev/hda
[ Utiliser la commande "p" pour afficher la table des partitions, "n"
  pour créer une partition et "w" pour écrire les modifications
  sur le disque. Un redémarrage peut s'avérer nécessaire. ]

# mkfs -t ext3 /dev/hda9
.../...
# mount -t ext3 /dev/hda9 /opt
# echo /dev/hda9 /opt ext3 defaults 1 2 >> /etc/fstab
```

Exemple de création d'un système de fichiers dans un « fichier image » :

```
# dd if=/dev/zero of=/var/local/myfs bs=1024 count=102400
.../...
# mkfs -t ext3 /var/local/myfs
[ Une confirmation est demandée car "/var/local/myfs" n'est pas un
  fichier spécial de type bloc. ]
.../...
# mkdir /mnt/tmp
# mount -o loop -t ext3 /var/local/myfs /mnt/tmp
# echo /dev/hda9 /opt ext3 loop 1 2 >> /etc/fstab
```

Réparer un système de fichiers

La commande `fsck` permet de vérifier l'intégrité d'un système de fichiers et, le cas échéant, d'effectuer une tentative de réparation. Le système de fichier ne doit pas être monter. Pour cette raison ces tests sont souvent effectués en *runlevel* 1 (*single*) :

```
# init 1
# umount /usr
# fsck -t ext3 -f /dev/hda7
# mount /usr
# init 5
```

Un autre moyen consiste à forcer la vérification de l'intégrité de tous les systèmes de fichiers locaux au démarrage du système. Pour cela, il suffit de créer un fichier vide nommé `forcefsck` dans la racine de l'arborescence :

```
# touch /forcefsck
# init 6
```

Les blocs de données orphelins retrouvés lors de la réparation de systèmes de fichiers *ext2* ou *ext3* sont placés dans les répertoire `lost+found` présents dans la racine de chaque système de fichiers *ext2* ou *ext3*.

Monter un système de fichiers réseau

Exemple d'un partage de fichiers Windows nommé « archives » sur un serveur « serveur » d'un domaine « WG » avec l'utilisateur « utilisateur » et le mot de passe « mot_de_passe ». Ce partage est attaché au point de montage `/mnt/archives` :

```
# mkdir /mnt/archives
# touch /etc/passwd.smb
# chmod 600 /etc/passwd.smb
# vi /etc/passwd.smb
[ Ajouter les lignes suivantes au fichier :
    username = utilisateur
    password = mot_de_passe
  Puis enregistrer. ]
# mount -t smb -o workgroup=WG,credentials=/etc/passwd.smb       \
    //serveur/archives /mnt/archives
# echo //serveur/archives /mnt/archives smb                      \
    workgroup=WG,credentials=/etc/passwd.smb 0 0 >> /etc/fstab
```

Gérer la mémoire virtuelle

Linux sait utiliser deux types de support pour gérer la mémoire virtuelle :

- des partitions ;
- des fichiers.

Les fichiers de mémoire virtuelle sont moins performants.
En général une partition est utilisée. Si le système contient deux disques ou plus, créer une partition dédiée à la mémoire virtuelle sur chacun d'entre eux peut améliorer les performances.

Ces partitions doivent avoir un identifiant système 0x82, il est nécessaire d'utiliser la commande « `t` » de l'outil `fdisk` de Linux pour modifier l'identifiant d'une nouvelle partition.

Créer un espace de mémoire virtuelle

Pour configurer un nouvel espace de mémoire virtuelle et l'activer de manière permanente, il est nécessaire de :

1) créer une partition ou un fichier vide ;
2) créer une signature de mémoire virtuelle avec la commande `mkswap` ;
3) utiliser la commande `swapon` pour activer l'espace de mémoire virtuelle ;
5) ajouter une ligne au fichier `/etc/fstab` afin que l'espace de mémoire virtuelle soit activée à chaque démarrage du système.

Voici un exemple pour ajouter un fichier de 100 Mo à la mémoire virtuelle :

```
# dd if=/dev/zero of=/swapfile bs=1024 count=102400
.../...
# mkswap /swapfile
Initialisation de la version de l'espace de swap 1, taille = 104853 kB
# swapon /swapfile
# echo /swapfile swap swap defaults 0 0 >> /etc/fstab
# swapon -s
Filename                Type            Size    Used    Priority
/dev/hda3               partition       524152  0       -1
/swapfile               file            102392  0       -2
```

Archives

`tar` (pour *Tape ARchiver*) est une commande Unix qui permet de gérer des archives d'ensemble de fichiers et/ou de répertoires. Son utilisation première était la manipulation de ces archives sur une cassette.
La commande `tar` est désormais très utilisée pour créer des fichiers d'archives véhiculés ensuite sur le réseau (comme une archive *zip*).

L'option « `c` » de tar est utilisée pour créer une archive. L'option « `f` » permet de spécifier la destination (fichier ou périphérique). Les exemples suivants créent tous deux une archive du répertoire `/home`, l'une dans un fichier, l'autre sur un lecteur de bande SCSI (DAT, DLT, etc.) :

```
# cd /; tar cf /mnt/backups/backup-home.tar home
# cd /; tar cf /dev/st0 home
```

Certaines versions de `tar` enregistrent le chemin absolu des fichiers archivés si celui-ci est indiqué. Attention lors de la restauration ! En général, on considère qu'il est sage d'archiver un répertoire en précisant son nom depuis son parent (chemin relatif).

Archives

La commande `tar` ne compresse pas les données. Il faut pour cela utiliser la commande `compress` :

```
# cd /; tar cf /mnt/backups/backup-home.tar home
# compress /mnt/backups/backup-home.tar
```

La commande `compress` remplace les fichiers donnés sur sa ligne de commande par des versions compressées de ceux-ci. L'extension « `.Z` » est ajoutée à leur nom.

Une autre manière de faire consiste à utiliser un tube :

```
# cd /; tar cf - home | compress > /mnt/backups/backup-home.tar.Z
```

(Le caractère « `-` » représente ici la sortie standard.)

Enfin, si la version de `tar` utilisée est celle du projet GNU, l'option « `Z` » permet de simplifier la ligne de commande :

```
# cd /; tar cZf /mnt/backups/backup-home.tar.gz home
```

`compress`, `gzip` et `bzip2`

La commande `compress` est ancienne et généralement utilisée par soucis de comptabilité. D'autres outils de compression tendent à se généraliser.

Le tableau suivant récapitule les caractéristiques des trois outils de compression les plus répandus dans le monde Unix :

commande compression	**commande décompression**	**efficacité**	**vitesse**	**extension générée**	**option GNU `tar`**
`compress`	`uncompress`	moyenne	moyenne	`.Z`	`Z`
`gzip`	`gunzip`	bonne	moyenne	`.gz`	`z`
`bzip2`	`bunzip2`	excellente	lente	`.bz2`	`j`

Archives

L'option « `t` » de la commande `tar` permet de lister le contenu d'une archive. L'option « `v` » permet d'accroître le niveau de détails affichés.

Par exemple :

```
# tar tvzf /mnt/backups/backup-home.tar.gz
.../...
```

Enfin, l'option « `x` » permet d'extraire le contenu d'une archive. Sans autre précision, le contenu de l'archive est restauré dans le répertoire courant si un chemin relatif a été utilisé lors de la création de l'archive ou dans le répertoire original si c'est un chemin absolu qui a été utilisé.

Par exemple :

```
# mkdir /home/tmp-restauration
# cd /home/tmp-restauration
# tar xzf /mnt/backups/backup-home.tar.gz
```

Modules du noyau

Certaines fonctionnalités du noyau Linux peuvent être compilées indépendamment de celui-ci. Elles sont compilées sous forme de module. L'utilisation des modules tend à se généraliser.

Avantages :

- réduction de la taille du noyau ;
- configuration dynamique du matériel ;
- possibilité d'ajout de certaines fonctionnalités sans recompiler le noyau ;
- des pilotes de périphériques peuvent être livrés sous forme de binaires.

Inconvénients :

- complexité accrue.

Les modules se présentent sous forme de fichiers binaires et sont installés dans le répertoire `/lib/modules/version_du_noyau`.

Modules du noyau

Un ensemble de commandes permettent de manipuler les modules. Pour afficher la liste de ceux qui sont actuellement chargés par le noyau :

```
# lsmod
Module                  Size  Used by
parport_pc             19392  1
lp                      8236  0
parport                29640  2 parport_pc,lp
autofs4                10624  0
sunrpc                101064  1
8139too                17792  0
.../...
```

Supprimer un module chargé :

```
# rmmod module
```

Charger un module :

```
# insmod /chemin/module.o [symbole=valeur ...]
```

Modules du noyau

Un module peut dépendre d'un ou plusieurs autres modules. Afin de charger un module et tous ceux qui lui sont nécessaires, on utilise la commande `modprobe` à la place de `insmod`. Un autre intérêt de `modprobe` est qu'il n'est pas nécessaire de lui donner le chemin complet d'accès au fichier du module.

Afin d'éviter de spécifier à chaque fois les paramètres des modules, ceux-ci sont stockés dans le fichier `/etc/modules.conf` (ou `/etc/modprobe.conf` pour les noyaux 2.6).

Si une fonctionnalité lui manque, le noyau Linux va essayer de combler cette lacune en recherchant un module adéquat. Il utilisera pour cela le contenu des fichiers `modules.conf` ou `modprobe.conf`.
Par exemple, une ligne de ce fichier est généralement destinée à configurer la carte réseau :

```
alias eth0 eepro100
```

Ainsi, si aucune carte réseau n'est reconnue par le noyau, la commande :

```
# ifconfig eth0 192.168.2.1
```

va provoquer la recherche d'un module `eth0` qui est un alias vers le nom du module réellement utilisé (`eepro100.o`).

Interfaces

L'élément de base de la configuration de réseau Linux est l'interface.

Un système Linux en possède généralement plusieurs.

Elles sont nommées en fonction de leur type et numérotées dans l'ordre d'activation des pilotes de périphériques et selon leur position sur le bus physique de données.

Les types d'interfaces les plus représentés sont :

- `lo` interface dite de *loopback* ;
- `ethn` interfaces Ethernet ;
- `pppn` liens PPP.

L'interface `lo` est toujours associée à l'adresse IP 127.0.0.1, la procédure d'installation du système se charge de réaliser cette association.

Les interfaces de type Ethernet sont configurées également par la procédure d'installation si elles ont été détectées.

Interfaces

Les pilotes de périphériques sont très souvent disponibles sous forme de modules du noyau. Ces modules peuvent être activés de deux manières différentes :

- le noyau charge le module de manière automatique lorsqu'une commande `ifconfig` fait référence à une interface qui n'est pas connue, un alias doit exister dans le fichier `/etc/modules.conf` :

```
alias eth0 3c59x
```

- un script charge explicitement le module lors du démarrage du système.

Si la procédure d'installation du système détecte une carte réseau, elle configure un alias automatiquement.

Les paramètres sont spécifiés dans le fichier `modules.conf` de cette manière :

```
alias eth0 ne
options ne io=0x220 irq=11
alias eth1 e1000
options e1000 Speed=1000 RxDescriptors=128
```

La commande `ifconfig`

La commande `ifconfig` est utilisée pour activer et configurer les interfaces.

L'option `-a` permet d'afficher la liste détaillée des interfaces réseau du système :

```
# ifconfig -a
eth0      Lien encap:Ethernet  HWaddr 00:20:ED:36:3C:EE
          inet adr:192.168.200.200  Bcast:192.168.200.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15356 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12392 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:100
          RX bytes:1745320 (1.6 Mb)  TX bytes:4365502 (4.1 Mb)
          Interruption:10 Adresse de base:0x5f00

lo        Lien encap:Boucle locale
          inet adr:127.0.0.1  Masque:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:929804 errors:0 dropped:0 overruns:0 frame:0
          TX packets:929804 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:0
          RX bytes:65622189 (62.5 Mb)  TX bytes:65622189 (62.5 Mb)
```

La commande `ifconfig`

Les interfaces sont activées ou désactivées comme ceci :

```
# ifconfig eth0 up
# ifconfig eth0 down
```

Enfin, l'association d'une configuration IP à une interface se réalise ainsi :

```
# ifconfig eth1 192.168.200.1 netmask 255.255.255.0
    broadcast 192.168.200.255
```

Les paramètres `netmask` et `broadcast` sont optionnels.

S'ils ne sont pas fournis, la valeur du premier est déduite à partir de la classe du réseau configuré (A, B ou C), le second prend la plus grande valeur dans l'espace d'adresses IP défini par l'adresse de l'interface et le masque de réseau.

Pour configurer plusieurs adresses IP sur une seule interface physique, il suffit de créer des alias en utilisant la notation « `ethn:a` » avec la commande `ifconfig`. Par exemple :

```
# ifconfig eth0:0 192.168.201.10
```

Les fichiers standards

Il existe un ensemble de fichiers standards qui décrivent la configuration de la couche IP du système :

`/etc/protocols`	protocoles IP ;
`/etc/services`	services TCP et UDP ;
`/etc/rpc`	services RPC (portmap) ;
`/etc/hosts`	noms statiques des systèmes ;
`/etc/networks`	noms statiques des réseaux.

Ces fichiers agissent comme des bases de données. Les trois premiers sont rarement modifiés, leur contenu est normalisé dans sa grande majorité.
Les deux derniers sont peu utilisés sur les réseaux qui opèrent un serveur de noms (DNS).

Fichiers de configuration des interfaces – Redhat

Les fichiers de configuration des interfaces réseau des distributions Redhat et assimilées (Fedora, CentOS, RHEL, etc.) forment un ensemble assez complexe.

Le fichier suivant contient les paramètres généraux :

```
/etc/sysconfig/network
   NETWORKING=yes
   HOSTNAME=jo.luke.net
   GATEWAY=192.168.200.1
```

Pour la configuration de l'interface `iface` du profile `profile`, les fichiers suivants sont recherchés, le premier trouvé est pris en compte :

```
/etc/sysconfig/networking/profiles/profile/ifcfg-iface
/etc/sysconfig/networking/default/ifcfg-iface
/etc/sysconfig/network-scripts/ifcfg-iface
   DEVICE=eth0
   BOOTPROTO=static
   BROADCAST=192.168.200.255
   IPADDR=192.168.200.200
   NETMASK=255.255.255.0
   NETWORK=192.168.200.0
   ONBOOT=yes
```

Fichiers de configuration des interfaces – Redhat

Seul le fichier `/etc/sysconfig/network-scripts/ifcfg-eth0` est créé durant la procédure d'installation du système.

Les fichier dans l'arborescence `/etc/sysconfig/networking` ne sont créés que lorsque l'utilitaire `redhat-config-network` a été utilisé.

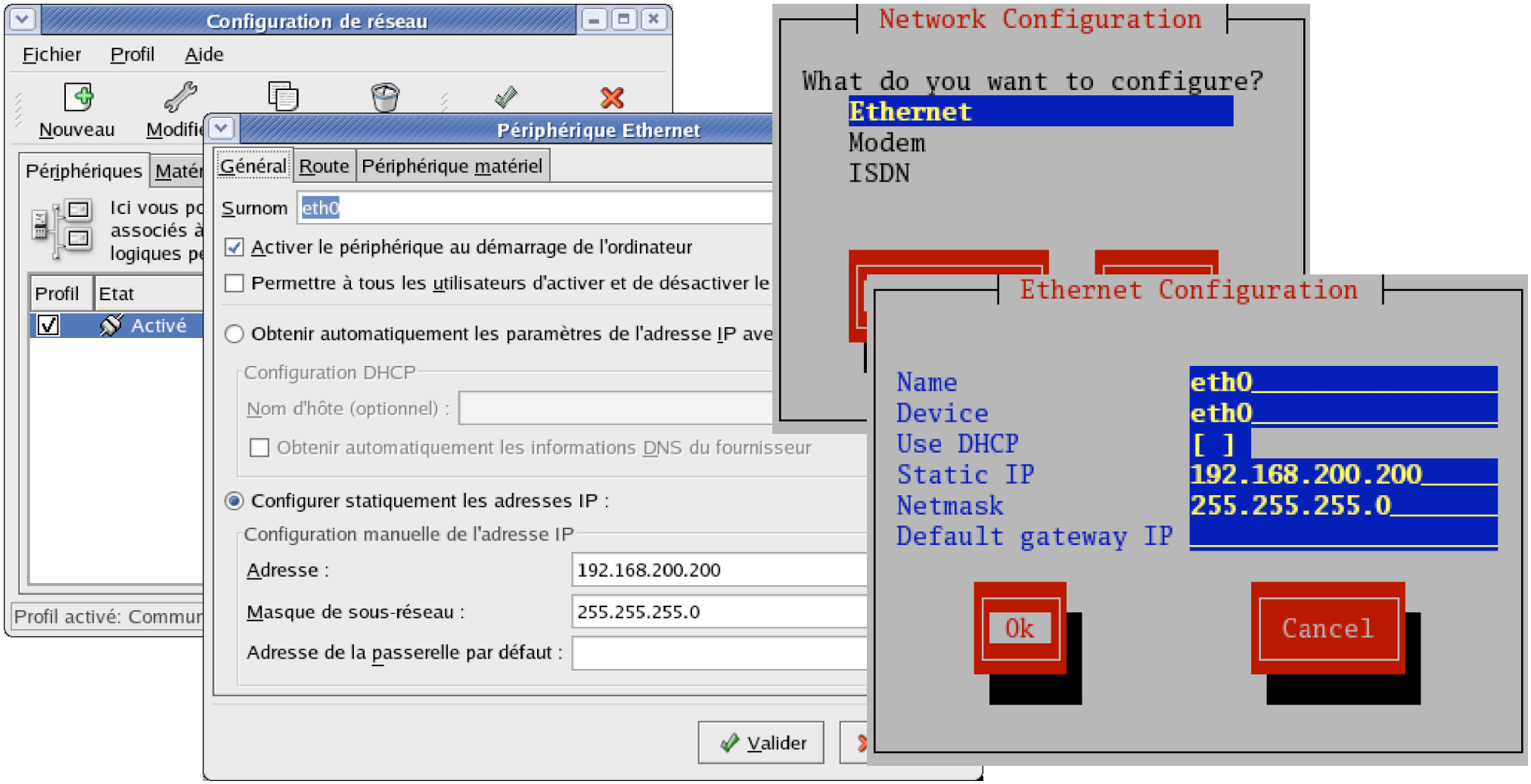
Le format de ces fichiers (et de tous ceux qui se trouvent dans le répertoire `/etc/sysconfig`) est détaillé dans ce document :

`/usr/share/doc/initscripts-7.14/sysconfig.txt`

(Le numéro de version du paquetage `initscripts` peut différer d'un système à l'autre.)

Fichiers de configuration des interfaces – Redhat

Voici des copies d'écran du mode graphique et du mode texte du programme `redhat-config-network` :



Fichiers de configuration des interfaces – Debian

Coté Debian, les choses sont plus simples.

Le fichier suivant contient les paramètres généraux :

```
/etc/network/options
   ip_forward=no
   spoofprotect=yes
   syncookies=no
```

Et celui-ci contient la configuration de toutes les interfaces :

```
/etc/network/interfaces

   auto lo
   iface lo inet loopback

   auto eth0
   iface eth0 inet static
       address 192.168.200.1
       netmask 255.255.255.0
       gateway 192.168.200.254
```

Le format de ce fichier est documenté dans la page de manuel interfaces(5).

Les commandes `ifup` et `ifdown`

Pour Redhat comme pour Debian :

- la commande `ifup` active ou réactualise la configuration d'une interface réseau ;
- la commande `ifdown` désactive une interface réseau.

Elles s'utilisent ainsi :

```
# ifup eth0
# ifdown eth0
```

Dans le cas de Debian, l'option `-a` permet de configurer toutes les interfaces réseau définies dans le fichier `/etc/network/interfaces` avec la directive `auto`.

Routes statiques

La gestion des routes se fait avec la commande `route`.

Pour afficher la table de routage statique, il est possible d'utiliser l'une de ces deux commandes :

```
# route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
192.168.200.0   *               255.255.255.0   U     0      0        0 eth0
default         lucky.luke.net  0.0.0.0         UG    0      0        0 eth0
# netstat -r
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic MSS Fenêtre irtt Iface
192.168.200.0   *               255.255.255.0   U       0 0          0 eth0
default         lucky.luke.net  0.0.0.0         UG      0 0          0 eth0
```

Les adresses IP ne seront pas résolues si l'option `-n` (commune aux deux commandes) est utilisée.

Routes statiques

Trois cas sont à considérer pour l'ajout de route :

- ajout d'une route vers une machine isolée :

```
# route add -host 10.20.30.1 gw 192.168.200.2
```

- ajout d'une route vers un réseau :

```
# route add -net 10.20.30.0 netmask 255.255.0.0 dev eth0
```

- ajout de la route par défaut :

```
# route add default gw gw.reseau.fr
```

La cible d'une route peut être une passerelle (`gw`) ou une interface (`dev`).
Dans le premier cas, l'adresse IP ou le nom de la passerelle peuvent être utilisés.

Voici comment supprimer une route :

```
# route del -net 10.20.30.0 netmask 255.255.0.0
```

Fichiers de configuration des routes – Redhat

Lorsque le programme `redhat-config-network` est utilisée, les routes statiques sont écrites dans le fichier :

```
/etc/sysconfig/networking/devices/eth0.route
   GATEWAY0=192.168.200.50
   NETMASK0=255.255.255.0
   ADDRESS0=192.168.201.0
   GATEWAY1=192.168.200.201
   NETMASK1=255.255.255.255
   ADDRESS1=192.168.202.1
```

Dans le cas contraire, il faut utiliser le fichier :

```
/etc/sysconfig/network-scripts/route-eth0
   192.168.201.0/24 via 192.168.200.50
   192.168.202.1/32 dev eth0
```

Attention, le format de ces fichiers diffère, les deux exemples présentés ci-dessus sont équivalents. (Le format utilisé dans le second cas est celui de la commande « `ip route` ».)

Fichiers de configuration des routes – Debian

Sous Debian les routes statiques doivent être configurées « à la main » dans le fichier `/etc/network/interfaces` :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 192.168.200.1
  netmask 255.255.255.0
  gateway 192.168.200.254
  up route add -net 192.168.201.0 netmask 255.255.255.0 gw 192.168.200.50
  up route add -host 192.168.202.1 dev eth0
  down route del -net 192.168.201.0 netmask 255.255.255.0
  down route del -host 192.168.202.1
```

Résolution des noms

La résolution des noms est réalisée via différents mécanismes qui sont listés dans les fichiers suivants :

```
/etc/host.conf
   order hosts,bind

/etc/nsswitch.conf
   passwd:     files
   shadow:     files
   group:      files
   hosts:      files dns
   bootparams: nisplus [NOTFOUND=return] files
   ethers:     files
   netmasks:   files
   networks:   files
   protocols:  files
   rpc:        files
   services:   files
   netgroup:   files
   publickey:  nisplus
   automount:  files
   aliases:    files nisplus
```

Résolution des noms

Le fichier de configuration pour la résolution des noms *via* DNS utilise le fichier `/etc/resolv.conf` :

```
search luke.net
nameserver 192.168.200.1
nameserver 192.168.200.10
```

Le paramètre `search` indique les domaines qui seront utilisés comme suffixes de recherche. Chacun des paramètres `nameserver` spécifie un serveur de noms DNS.

Le fichier `/etc/hosts` contient des définitions de noms statiques :

```
127.0.0.1         localhost.localdomain  localhost
192.168.200.201   jack.luke.net  jack
192.168.200.1     gw
```

Les super-serveurs inetd et xinetd

Le super-démon inetd écoute un certain nombre de ports TCP ou UDP. Lorqu'une connexion est activée sur l'un de ces ports, il exécute une application.

Cela évite d'avoir un nombre important de démons, mais le temps de réponse est plus long. C'est pourquoi les démons servants les services très utilisés (HTTP, SMTP, DNS, etc.) sont exécutés au démarrage du système, les autres sont activés par inetd.

Le fichier de configuration d'inetd est `/etc/inetd.conf`.

Le démon xinetd est de plus en plus utilisé à la place d'inetd, il utilise le fichier de configuration `/etc/xinetd.conf` et le répertoire `/etc/xinetd.d`.

Cependant, d'une manière générale, ces démons sont de moins en moins employés car beaucoup de services qu'ils prenaient en charge sont abandonnés (finger, talk, etc.) ou exécutés sous forme de démons sur des systèmes spécialisés (POP, SMTP, HTTP, whois, etc.).

Relais de messagerie minimal – Postfix

Les systèmes Unix communiquent beaucoup par le biais de la messagerie électronique. Par exemple, le résultat des commandes exécutées par cron parviennent aux utilisateurs par ce canal.

Il est donc nécessaire de réaliser une configuration minimale du MTA.

Nous nous limiterons ici à la configuration d'un relais simple (qu'on appelle généralement « *smart host* »).

Dans le cas de Postfix, il suffit de positionner le paramètre `relayhost` du fichier `/etc/postfix/main.cf` à la valeur adéquate :

```
relayhost=[smtp-gw.domaine.fr]
```

Relais de messagerie minimal – Sendmail

Dans le cas de Sendmail, deux cas peuvent se présenter :

- la configuration est gérée par *m4* (fichier `/etc/mail/sendmail.mc`) ;
- le fichier `/etc/mail/sendmail.cf` est utilisé directement.

Dans le premier cas, il faut :

1) définir la variable `SMART_HOST` du fichier `/etc/mail/sendmail.mc` :

```
define(`SMART_HOST',`[smtp-gw.domaine.fr]')
```

2) régénérer le fichier `sendmail.cf` à partir de `sendmail.mc` :

```
# make -C /etc/mail
```

(Attention au sens des apostrophes !)

Dans le second cas, rechercher la ligne du fichier `/etc/mail/sendmail.cf` qui commence exactement par les deux caractères « `DS` » et y ajouter à suivre (sans espaces) la valeur adéquate :

```
# "Smart" relay host (may be null)
DS[smtp-gw.domaine.fr]
```

Outils

Les outils suivants peuvent s'avérer utiles lors de la mise au point d'une configuration réseau sous Linux :

- Tester la connectivité IP :

```
# ping jo.luke.net
```

- Lister les ports Unix, TCP et UDP en écoute :

```
# netstat -lp
```

- Observer la route d'un système à l'autre :

```
# traceroute jo.luke.net
```

- Résoudre des noms de réseau avec DNS :

```
# host lucky.luke.net
# host -t mx luke.net
# host 192.168.200.1
```

Références

Livres

« Administration Linux à 200% » – Rob Flickenger, collectif – O'Reilly
« Le système Linux » – Matt Welsh, Matthias Kalle Dalheimer, Terry Dawson et Lar Kaufman – O'Reilly
« Les bases de l'administration système » – Æleen Frisch – O'Reilly
« Administration réseau sous Linux » – Olaf Kirch et Terry Dawson – O'Reilly

Sites Web

« Linux France » – `http://www.linux-france.org`
Le site de la société Redhat – http://www.redhat.fr
Le site du projet Fedora – `http://fedora.redhat.com`

Autres

Le forum fr.comp.os.linux sur USENET
« Linux Magazine », mensuel