Gestion évoluée des services sous Linux

Article publié dans GNU/Linux Magazine France, numéro 65 (Octobre 2004).

Les phases critiques de démarrage ou d'arrêt des services (généralement de simples processus lancés en tâche de fond) ont toujours été un point clef dans l'évaluation du critère de maintenabilité d'un système. Linux n'échappe pas à cette règle et l'on est en droit de s'attendre à ce que certaines facilités aient été développées afin de minimiser et simplifier cette tâche pour l'administrateur.

La problématique liée à la gestion des services est en effet largement complexifiée par les différentes dépendances qui existent entre ces services : Il paraît en effet inconcevable de lancer un serveur Apache avant l'activation du réseau tout comme il paraît absurde de vouloir activer un pare-feu bien après celui traitant de la connexion Internet. Un service ne doit pas être inséré n'importe où dans le processus d'amorçage même si certaines libertés peuvent être prises dans l'ordre de démarrage de certains services (principalement réseaux).

Cette réalité s'applique aussi dans la phase d'arrêt des services et lors du passage entre les différents niveaux d'exécutions (runlevels).

Il paraît donc intéressant de s'attarder un peu sur le sujet afin d'évaluer ce que nous propose les distributions Linux généralistes majeures que l'on peut trouver sur la scène européenne : J'ai nommé Mandrake, Redhat, SuSE et enfin Debian.

Autant vous prévenir tout de suite, les différentes solutions proposées ne sont pas très homogènes : Un administrateur devra généralement jongler avec plusieurs utilitaires avant de pouvoir gérer correctement des services sur un parc hétérogène de distributions Linux.

Mais avant d'aller plus loin, il semble indispensable de réviser un peu notre sujet.

On trouve deux techniques principales pour initialiser un système Unix (dont Linux est un proche cousin) : L'initialisation de type system V ou BSD.

Linux a adopté le principe mis en place par system V (plus puissant et évolutif que l'init de type BSD) : Chaque service est associé à un script unique qui est placé dans un répertoire commun (dans notre cas, **/etc/init.d**), le démarrage et l'arrêt des services se faisant par l'intermédiaire d'une collection de liens symboliques placés dans les répertoires présents sous **/etc/rc[0-6].d**

Selon le runlevel sélectionné dans la phase d'amorçage du système, le processus init (père de tous les processus de votre système) va se placer dans le répertoire correspondant, par exemple / etc/rc5.d dans le cas du runlevel 5, et va exécuter tous les scripts (en fait les liens symboliques) commençant par K avec le paramètre « stop » (K pour Kill) pour ensuite enchaîner sur l'exécution de tous les scripts commençant par S avec le paramètre « start » (S pour Start).

Le nommage des scripts est libre même s'il est fortement conseillé de rester proche de ceux des services qu'il sont sensés placer en tâche de fond : Cela facilitera grandement la tâche d'administration de la machine.

Le nommage des liens symboliques est plus complexe : La première lettre doit correspondre au type du lien (S pour Start ou K pour Kill) suivi d'un numéro sur 2 chiffres correspondant à l'ordre de prise en compte (un service portant le numéro 12 sera prioritaire au démarrage sur un service portant le numéro 30) et terminé par le nom du service (de préférence).

Par exemple, le service permettant d'activer le réseau sur la Mandrake est habituellement placé sous **/etc/init.d/network** alors que le lien symbolique est disponible sous **/etc/rc5.d/S10network** dans le cas du runlevel 5. Un runlevel qui ne dispose pas du service réseau se verra attribuer un lien symbolique K90network permettant l'arrêt du réseau dans le cas d'un changement de runlevel (le numéro d'ordre est en général symétrique car ce qui est démarré en premier doit être généralement arrêté en dernier).

A ce stade, vous vous dites peut être que les choses sont claires et que, finalement, la gestion de ces liens symboliques ne va pas être si compliqué : Pourquoi donc ne pas directement les créer à la

main ?

Si l'on y regarde de plus prêt, on s'aperçoit que les choses ne sont aussi simples.

Regardons le cas de la SuSE : On creusant un petit peu le processus d'activation des services (man init.d), on constate que celui-ci diverge légèrement de ses congénères sur certains aspects que nous aurions pu espérer génériques. Il s'avère que les liens symboliques ne sont pas disposés de la même façon dans les différents runlevels : Les liens symboliques d'arrêt et de démarrage d'un service cohabitent en effet dans chaque répertoire **rc[x].d** ou le service est activé. Le principe utilisé est le suivant : Pour éviter les démarrages redondants en cas de changement d'un niveau d'exécution, seuls les services qui n'ont pas de liens start dans le précédent niveau d'exécution sont démarrés. Par analogie et pour éviter les arrêts redondants, seuls les services qui n'ont pas de liens start dans le niveau d'exécution courant sont stoppés.

Ce n'est certes pas fondamental, mais on comprend maintenant pourquoi il est important de disposer d'outils simplifiant les étapes de création et de suppression de ces liens symboliques : Leurs gestions étant une tâche on ne peut plus fastidieuse qui se doit d'être minimisée au maximum et qui peut éventuellement varier d'une distribution à l'autre (la preuve en est faite au moins au niveau de la SuSE).

Gestion des services sous Debian

Les utilitaires qui permettent la gestion des services sous la Debian s'appellent **invoke-rc.d** et **update-rc.d**. Le premier sert à manipuler un service (généralement le démarrer ou l'arréter) alors que le second sert à interroger la configuration des services et leurs comportements au démarrage.

Traitons tout d'abord le cas du programme **update-rc.d** qui permet de mettre en place ou de supprimer les liens symboliques de démarrage et d'arrêt des services de type system V.

L'approche Debian considère que les informations telles que la priorité d'arrêt et de démarrage du service, ainsi que la liste des niveaux d'exécution par type de lien symbolique (arrêt ou démarrage) doivent être précisées directement dans la ligne de commande : Aucune facilité n'est offerte afin de stocker ces informations directement dans le script.

Avant d'appeler la commande **update-rc.d**, il faut s'assurer que le service (en fait le script) a bien été copié dans le répertoire */etc/init.d*, sinon elle n'aura aucun effet.

Par exemple, si vous vouliez activer un service quelconque avec un niveau de priorité 20 pour le démarrage, 80 pour l'arrêt et ce dans les runlevels 3 et 5, il vous faudra saisir sous root la commande suivante :

```
# update-rc.d <service> start 20 3 5 . stop 80 0 1 2 4 6 .
```

Cette commande un peu étrange nécessite quelques explications.

On décompose la commande en trois parties :

- On précise d'abord le nom du script présent sous le répertoire */etc/init.d* : <service>,
- On indique ensuite la priorité des liens symboliques de type démarrage du service (préfixé par S) avec la liste des niveaux d'exécutions terminée par un point : « start 20 3 5 . »,
- On indique enfin la priorité des liens symboliques de type arrêt du service (préfixe par K) avec avec la liste des niveaux d'exécutions terminée par un point : « stop 80 0 1 2 4 6 . ».

Comme nous l'avons vu dans l'introduction, les liens symboliques permettant l'arrêt du service ne sont pas disposés dans les mêmes runlevels que les liens symboliques activant le service sachant que les runlevels 0123456789S sont supporté par le processus init Debian.

Il est possible de disposer d'une configuration par défaut en utilisant le mot-clef **default** :

```
# update-rc.d <service> default
```

Dans ce cas, la configuration est équivalente à : **start 20 2 3 4 5 . stop 20 0 1 6 .**

Supprimer un service devient trivial en appelant la commande :

```
# update-rc.d <service> remove
```

On notera la disponibilité des options [-f] et [-n] permettant successivement de forcer la suppression des liens symboliques (si le script est toujours présent sous /etc/init.d) et de simuler ce que va faire la commande (très pratique pour la prise en main sans risque de cette commande).

La commande est simple mais semble avoir mal vieillie lorsque l'on commence à regarder un peu ce qui ce fait du coté de la « concurrence » ; en particulier, parce qu'elle ne pas permet d'obtenir une synthèse rapide de l'état des services qui tournent sur une machine.

La commande **invoke-rc.d**, quand à elle, permet d'appeler le script gérant un service avec ses options comme par exemple start, stop restart ou status (si celle dernière option est disponible). Le nom du service correspond en fait à celui présent dans le répertoire */etc/init.d*.

Par exemple, pour démarrer un service :

```
# invoke-rc.d <service> start
```

Ou encore pour le relancer :

```
# invoke-rc.d <service> restart
```

Vous noterez que vous aurez obtenue le même résultat en appelant directement le script gérant le service comme dans l'exemple suivant permettant de relancer le démon SSH :

```
# /etc/init.d/sshd restart
```

Par contre, il n'est pas possible de manipuler les services gérés par les super-serveurs **inetd** ou **xinetd** par l'intermédiaire de ces deux commandes.

Gestion des services sous Redhat et Mandrake

Initialement, la Mandrake est un fork de la Redhat : Il est donc logique de constater que ces deux distributions continuent de partager les mêmes utilitaires pour gérer les services, en l'occurrence **chkconfig** et **service**.

Service est similaire à la commande **invoke-rc.d** de la Debian de la même façon que **chkconfig** peut être considéré comme un proche cousin de **update-rc.d** : L'analogie s'arrête à une vague ressemblance. Nous allons voir que les outils fournis par Redhat offrent bien plus de fonctionnalités.

La commande **service** permet de manipuler un service courant mais offre surtout une réelle couche d'abstraction pour la manipulation des services gérés par **inetd** ou **xinetd**. L'accès est transparent pour l'utilisateur.

Par exemple, si je veux savoir comment est géré le service ftp (que je sais géré par **xinetd**) :

```
# service ftp
I need an action
ftp is a xinetd service
```

A savoir que les services gérés sous xinetd peuvent seulement être relancés (en considérant qu'ils sont déjà démarrés) :

```
# service ftp start
ftp is a xinetd services
Rechargement de la configuration :                         [  OK  ]
```

En interrogeant un service indépendant (livré sous forme d'un script) on obtient la liste des options disponibles avec lesquelles on pourra appeler le script, ce qui est extrêmement pratique :

```
# service network
I need an action
Utilisation : network {start|stop|restart|reload|status}
```

Par contre, si l'on cherche à interroger un service géré sous **xinetd** qui n'est pas encore démarré, vous aurez l'agréable surprise de voir s'afficher le message suivant :

```
# service fam
I need an action
fam is a xinetd service and it is disabled
to activate it do the following command:
chkconfig fam on
```

Cela nous permet de rebondir vers la commande **chkconfig** qui justement va nous permettre de gérer (on démarre ou on arrête un service) et de décider de la politique d'activation des services au démarrage du système.

Historiquement, c'est SGI qui a conçu le programme **chkconfig** qui était livré en standard dans leur système d'exploitation Unix IRIX : Les transfuges de ce système ne seront donc pas complètement dépaysés. Redhat a repris l'idée quelques années plus tard et a réalisé une version similaire en C qui est proposée par défaut sous la Redhat et la Mandrake (plus précisément, la version 1.3.6).

Une version dérivée de cet outil est même proposée sur le site [1] pour ceux qui sont intéressés par sa compilation sur d'autres Unix, la version que propose Redhat étant logiquement ciblée sur leur système d'exploitation maison.

Comment fonctionne **chkconfig** ? Cette commande permet d'obtenir la liste de tous les services actifs ou inactifs présents sur une machine, et cela par niveaux d'exécutions.

L'administrateur précise en ligne de commande la liste des niveaux d'exécutions où il souhaite démarrer et arrêter les services même si une configuration par défaut peut être extraite directement du script.

Pour chaque service que l'on souhaite gérer par l'intermédiaire de **chkconfig**, on pourra donc ajouter dans le script la liste des niveaux d'exécutions par défaut (pour éviter d'avoir à les préciser manuellement) ainsi, et c'est le plus important, que les priorités de démarrage et d'arrêt qui seront utilisées pour nommer les liens symboliques (ce dernier point étant obligatoire).

Car nous touchons là à une des limites de cet outil dont la prise en main est pourtant assez conviviale : Il faut obligatoirement préciser dans le script les niveaux de priorité pour l'arrêt et le démarrage du service. Cela ne pose généralement pas de problèmes pour les services fournis en standards dans ces deux distributions car ils comportent toujours les fameuses informations : Cela devient plus problématique lorsque vous souhaitez déployer un service un petit peu exotique entre plusieurs plateformes Linux car non seulement il vous faudra calculer vous même les niveaux de priorités de votre service (celui-ci peut nécessiter le démarrage préalable du serveur apache et donc ne doit pas être inséré n'importe où), mais il vous faudra aussi tenir compte des spécificités qui existent entre ces différents environnements. La chose peut donc rapidement tourner au casse-tête.

Malgré cela, l'outil reste agréable et très pratique : Les commandes sont intuitives et se mémorisent facilement ; il reste un outil de choix dans la boite à outil de tout administrateur.

Liste des fonctionnalités disponibles à travers **chkconfig** :

```
# chkconfig
chkconfig version 1.3.6 - Copyright (C) 1997-2000 Red Hat, Inc.
Ce produit peut être librement distribué selon les termes de la licence publique GNU (GPL).
utilisation : chkconfig --list [nom]
              chkconfig --add <nom>
              chkconfig --del <nom>
              chkconfig [--level <niveaux>] <nom> <on|off|reset>)
```

Pour obtenir l'état des services par niveau d'exécution :

```
# chkconfig --list
network    0:Arrêt  1:Arrêt  2:Marche  3:Marche  4:Marche  5:Marche  6:Arrêt
xinetd     0:Arrêt  1:Arrêt  2:Arrêt   3:Arrêt   4:Marche  5:Marche  6:Arrêt
sshd       0:Arrêt  1:Arrêt  2:Marche  3:Arrêt   4:Marche  5:Marche  6:Arrêt
[...]
Services basés sur xinetd :
        [...]
        ftp:    Marche
```

Ou celui d'un service en particulier :

```
# chkconfig --list sshd
sshd      0:Arrêt  1:Arrêt  2:Marche  3:Arrêt  4:Marche  5:Marche  6:Arrêt
# chkconfig --list ftp
        ftp:    Marche
```

Pour ajouter un service en suivant le paramétrage par défaut :

```
# chkconfig --add ftp
```

Pour supprimer un service :

```
# chkconfig --del ftp
```

Ou encore, pour activer un service au sein de différents niveaux d'exécutions (et donc ne pas utiliser la configuration proposée par défaut) :

```
# chkconfig --del sshd
# chkconfig --level 35 sshd on
# chkconfig --list sshd
sshd      0:Arrêt  1:Arrêt  2:Arrêt  3:Marche  4:Arrêt  5:Marche  6:Arrêt
```

Sachant que pour chaque service devant être géré par **chkconfig**, il faudra que le script comporte au moins les deux lignes de commentaires suivantes :

- Un premier commentaire précisant les niveaux d'exécutions et les priorités d'activation du service,
- Un second donnant une petite description du service (vous remarquerez l'utilisation du caractère # synonyme de commentaire qui ne doit pas être confondu avec le prompt shell des exemples précédents).

```
# chkconfig: 2345 20 80
# description: ce service sera démarré par défaut dans les\
# runlevels 2,3,4 et 5 avec une priorité de 20 au démarrage\
# et de 80 à l'arrêt. La description du service peut occuper\
# plusieurs lignes en utilisant le caractère « \ » en fin de ligne
```

Vous pouvez utiliser le caractère « - » en lieu et place de la liste de niveaux d'exécutions afin d'éviter de démarrer le service par défaut.

Le problème qui se pose à nous est le suivant : Les niveaux de priorités de démarrage et d'arrêt des services sont contenus directement dans les scripts ce qui peut nécessiter une petite intervention manuelle pour modifier l'ordre dans lequel on souhaite que le service soit démarré ou arrêté.

On se prend donc à rêver d'un utilitaire similaire qui nous permettra de ne pas avoir à préciser littéralement les priorités de démarrage ou d'arrêt (priorités absolues) mais qui les calculera automatiquement suivant les dépendances du service vis à vis d'autres services (priorités relatives).

Pourtant, un tel outil existe et nous allons voir que sa similitude avec **chkconfig** est assez frappante.

Gestion des services sous la SuSE

Depuis la version 7.1, la distribution SuSE a adopté les principes de gestion des services init.d proposés par la spécification LSB (Linux Standard Base) [2]. Cette spécification préconnise de maintenir à l'intérieur du script toutes les informations utiles concernant les dépendances entre les services (ce qui est nouveau) ainsi que la classique liste des niveaux d'exécutions (de façon assez similaire à ce que nous avons vu pour l'utilitaire **chkconfig**).

La nouveauté concerne donc la notion de dépendance entre les services qui va nous permettre de ne plus choisir manuellement et de façon arbitraire le numéro de priorité de démarrage ou d'arrêt d'un service (qui peut-être une source d'erreurs), mais bien de le faire calculer à un programme tiers en tenant compte de la liste des services déjà présents dans le système.

Le programme qui va nous permettre d'interpréter ces données et de mettre en place les différents liens symboliques dans notre arborescence système s'appelle **insserv** : A ma connaissance, il s'agit d'un programme développé en interne par SuSE (certainement au sein du regroupement UnitedLinux) et dont le nom et la fonction peut sans doute varier sous d'autres distributions supportant la spécification LSB.

Par exemple, on va pouvoir préciser que le service tourne par défaut dans les niveaux d'exécutions 35 et nécessiter le démarrage préalable du réseau : Quelque soit la configuration de la machine sur laquelle vous vous trouverez, les niveaux de priorités seront donc calculés en conséquence ce qui offre une souplesse incomparable.

Un script init.d qui souhaite utiliser toutes ces facilités doit proposer un en-tête normalisé qui commence par la ligne « ### BEGIN INIT INFO » pour se finir par la ligne « ### END INIT INFO » : Toutes les lignes de cet en-tête débutent par le caractère # sur la première colonne pour pouvoir être traitées par le shell comme un seul et unique commentaire.

Par convention, la spécification LSB impose l'utilisation d'un mot-clef normalisé pour chaque type d'informations (qui sera lui même suivi de un ou plusieurs arguments) :

```
### BEGIN INIT INFO
# Provides: fonction [fonctions ...]
# Required-Start: fonction [fonctions ...]
# Required-Stop: fonction [fonctions ...]
# Defaut-Start: runlevel [runlevels ...]
# Defaut-Stop: runlevel [runlevels ...]
# Short-Description: commentaire sur une ligne
# Description: commentaire sur plusieurs lignes
### END INIT INFO
```

Chaque mot-clef décrit un point fonctionnel précis utilisé par le programme insserv (l'ordre d'apparition des mots-clefs est indifférent dans l'en-tête) :

Provides	On précise un nom arbitraire qui symbolise la fonction du script : On utilise généralement le nom du service à démarrer pour des raisons évidentes de maintenabilité (il semble possible de préciser plusieurs fonctions) A noter que si plusieurs services proposent la même fonction, par exemple la fonction SMTP dans le cas des démons postfix et sendmail, vous devrez obligatoirement utiliser le même nom
Required-Start	Ce mot-clef est essentiel car il permet de lister le nom des services (au sens du mot-clef **Provides**) qui devront obligatoirement être démarrés avant de pouvoir activer notre service Cette information sera utilisée par le programme insserv pour calculer les dépendances entre tous les scripts et s'assurer que ceux-ci seront exécutés dans un ordre correct
Required-Stop	La spécification LSB nous permet de donner une liste de services qui devront toujours être actifs avant l'arrêt de notre propre service : Ce mot-clef est actuellement ignoré par la SuSE pour les raisons que nous avons exposées au début de cet article
Default-Start	On précise la liste des niveaux d'exécutions dans lesquels devront être démarrés le service, en l'occurrence le nom du script avec l'argument « start »
Default-Stop	Dans le cas de la SuSE, ce mot-clef n'est pas interprété
Short-Description	On donne une petite description du service qui doit tenir sur une ligne
Description	On donne une description plus longue du service qui peut tenir sur plusieurs lignes : Chaque ligne supplémentaire doit débuter par le caractère # suivi d'une tabulation, ou encore le caractère # suivi d'au moins deux espaces. La description se termine lorsque ce critère n'est plus respecté

Dans un soucis de normalisation, la spécification LSB impose la déclaration de certaines fonctions considérées comme indispensables au bon amorçage du système et au calcul des dépendances entre les scripts : Ces fonctions normalisées commencent donc par le caractère $ afin de permettre de séparer l'espace de nommage entre les fonctions systèmes et les fonctions applicatives. Par analogie avec le shell, on peut parler de variables puisque le nom réel du ou des services est assigné ailleurs.

Le fichier **/etc/insserv.conf** permet justement de faire le lien entre ces variables et les fonctions réelles qui peuvent assez logiquement varier entre les systèmes (soit par leurs noms, soit par le nombre de services qu'ils peuvent remplacer).

La spécification LSB impose la déclaration des variables suivantes :

$local_fs	Implicite que tous les systèmes de fichier locaux à la machine sont montés
$network	On démarre les services réseaux de bas niveau, généralement l'initialisation des cartes réseaux
$named	Démarrage des démons qui s'occupent de la gestion du service de résolution de noms (si applicable), en général le démon named pour le logiciel BIND
$portmap	Démarrage des démons fournissant les services SunRPC utilisés par exemple dans le système de fichier réseau NFS (si applicable)
$remote_fs	Montage de tous les systèmes de fichier distants ainsi que tous les systèmes de fichier local (en l'occurrence, ceux de la fonction **$local_fs**)
$syslog	Démarrage du journal système permettant de tracer tous les évènements, généralement disponible sous **/var/log/messages**
$time	On démarre un gestionnaire d'horloge comme ntp, rdate ou encore l'horloge temps réel du système

La distribution SuSE proposait initialement la variable **$netdaemons** à la place de **$portmap** en élargissant le périmètre aux super-serveurs **inetd** ou **xinetd**, ce qui n'était pas forcément une mauvaise idée : Cela n'étant pas couvert par la spécification LSB, cette variable doit désormais être considérée comme obsolète.

Pour activer un service par l'intermédiaire de **insserv** en utilisant la configuration présente dans le script :

```
# insserv sshd

# chkconfig --list sshd
sshd                    0:off   1:off   2:off   3:on    4:off   5:on    6:off
```

Pour supprimer un service :

```
# insserv -r sshd

# chkconfig --list sshd
sshd                    0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

Pour activer un service en précisant dans la ligne de commande la liste des niveaux d'exécutions pour s'affranchir de celle spécifiée en dur dans le script :

```
# insserv sshd,start=5

# chkconfig --list sshd
sshd                    0:off   1:off   2:off   3:off   4:off   5:on    6:off
```

Vous devez sans doute être surpris que je continue d'utiliser la commande **chkconfig** alors que je me trouve désormais sous la SuSE. Ce sera donc une heureuse surprise car cette commande existe effectivement sur cette plateforme même s'il ne s'agit pas littéralement de l'utilitaire proposé par Redhat : Au lieu d'utiliser le formalisme de paramétrage des services communs aux plateformes Redhat/Mandrake que nous avons balayé dans la section précédente, la commande de la SuSE utilise le formalisme proposé par la spécification LSB par l'intermédiaire de notre utilitaire **insserv**.

Les personnes intéressées pourront constater que l'outil **chkconfig** est en fait un script perl qui encapsule des appels à insserv pour les phases d'ajout et de suppression d'un service : Malgré leurs similitudes, gardez à l'esprit que ce sont deux outils distincts dont les options ne sont pas toujours similaires.

Par contre, l'absence de **service** sur la SuSE est remarquée même si on peut faire l'équivalent à travers les commandes **insserv** et **chkconfig** : On aurait aimé conservé une commande unique entre toutes les plateformes.

Petit tableau récapitulatif de l'appel à la commande **chkconfig** sur plateforme SuSE :

Obtenir la liste et l'état de synthèse de tous les services dans le runlevel courant	`# chkconfig`
Obtenir la liste des services et leurs états d'activation pour tous les niveaux d'exécutions	`# chkconfig --list`
Obtenir l'état d'un service en particulier pour tous les niveaux d'exécutions	`# chkconfig --list <service>`
Désactiver un service	`# chkconfig <service> off` `# chkconfig --del <service>`
Activer un service en utilisant le paramétrage par défaut	`# chkconfig --add <service>` `# chkconfig <service> on`
Activer un service en précisant dans la ligne de commande les niveaux d'exécutions	`# chkconfig --set sshd 35` `# chkconfig sshd 35`

Comme vous pouvez le constater, les différences majeures concernent la façon dont nous allons préciser les niveaux d'exécutions où nous souhaitons activer le service. Le reste est assez proche.

Une fonctionnalité intéressante qui n'est disponible que sur la SuSE nous permet de sauvegarder le contexte des services dans un fichier pour pouvoir le rejouer plus tard : Cela peut permettre rapidement, et en une seule commande, d'appliquer sur un parc de machines la même configuration au niveau des services (et éviter ainsi des manipulations fastidieuses) :

```
# chkconfig > services.lst
# chkconfig -s < services.lst
```

Vous noterez enfin que même si l'utilitaire **insserv** n'est pas disponible sur votre distribution préférée, la spécification LSB impose de mettre à disposition un programme permettant d'installer et de désinstaller un script respectant le formalisme LSB : Les programmes sont nommés **install_initd** et **remove_initd**, et doivent être accessibles dans le répertoire **/usr/lib/lsb**.

Pour ajouter un service LSB :

```
# /usr/lib/lsb/install_initd <service>
```

Pour supprimer un service LSB :

```
# /usr/lib/lsb/remove_initd <service>
```

La distribution SuSE propose bien évidemment ces deux commandes en standard, au même titre que les distributions Redhat et Mandrake moyennant l'installation de paquets RPM optionnels. A ma connaissance, la distribution Debian ne propose pas le support LSB.

Une petite conclusion

Que pouvons nous en déduire ? Que la distribution SuSE a eu le bon goût de s'approprier le formalisme proposé par la spécification LSB dans la gestion des services même au sein de ses propres scripts init.d (ce qui n'est pas imposé par la norme) et que l'utilisation de la commande **chkconfig** permet dans la majorité des cas d'interroger et de gérer convenablement un parc hétérogène de machines Linux (beaucoup de distributions sont dérivées plus ou moins directement de la Redhat).

Le support de la spécification LSB offre la possibilité de déployer des services multi-plateformes ce qui est un gain appréciable (en partant du principe que les commandes **install_initd** et **remove_initd** sont disponibles sur les machines).

On s'habitue rapidement à la commande **service** proposée par Redhat et on regrette qu'elle ne soient pas disponible sur toutes les plateformes, en particulier sur la SuSE.

Le cas Debian est à part : En proposant des commandes certes efficaces mais un peu sommaires,

celle-ci se place un peu en marge. Sans vouloir heurter les puristes, il est dommage que cette solide distribution n'offre pas de facilités de type **chkconfig** ou **insserv** autorisant l'administrateur à se concentrer sur d'autres parties tout aussi critiques de son système.

Quid de la convergence ? Celle-ci pourrait facilement se faire autour d'un seul et même standard : La spécification LSB ; du moins en ce qui concerne la gestion des services. Tous les outils existent et sont sous licence libre, le reste est un simple travail d'intégration et de changement des habitudes.

Offrir un même jeu de commande, comme par exemple **chkconfig** et **service**, serait aussi un geste très apprécié et améliorerait la maintenabilité des systèmes (cela se joue parfois à quelques détails).

Une telle standardisation pourrait séduire les décideurs pressés et accélérer la pénétration de Linux au sein des entreprises ; elles pourraient aussi éviter aux administrateurs de s'arracher les cheveux à essayer de jongler entre plusieurs commandes pour une tâche aussi triviale que la gestion et le déploiement de services sous Linux.

Qu'en pensez vous ?

Lionel Tricon

Références :	
[1]	Chkconfig : http://www.fastcoder.net/~thumper/software/sysadmin/chkconfig/
[2]	Linux Standard Base : http://www.linuxbase.org/