



Remplacer LILO par GRUB

Le 2 août 2008 à 02:07.

je ne sais plus à partir de quelle version exactement la Mandriva a définitivement basculé sous GRUB, mais aujourd'hui c'est sur cette distribution le chargeur d'amorçage par défaut au détriment de Lilo. Maintenant lorsque vous avez passé plusieurs années à mettre un système à jour d'une version à l'autre sans jamais réinstaller, il se peut que vous ayez encore un LILO qui traîne. Voici donc comment s'en débarrasser.

Anatomie d'un MultiBoot

Pour bien comprendre l'intérêt de GRUB par rapport à LILO, il est nécessaire d'en connaître un peu plus sur le processus de démarrage d'un PC, et de Linux en particulier;

Lorsqu'un PC de base démarre, le BIOS, après avoir fait tous ses tests, va charger en mémoire le premier secteur (512 octets) du disque de démarrage appelé le MBR (master boot record) et en exécute le code. Ce dernier va analyser la table des partitions (contenu sur le même secteur) et déterminer si l'une d'elles est marquée amorçable. S'il en trouve une, il charge son premier secteur (secteur de démarrage ou boot sector) en mémoire et en exécute le code. Ce code là est spécifique à un chaque OS. Sous Windows par exemple, il va entraîner le chargement et l'exécution de NTLDR.

Dans le cas de Linux, le MBR contient le programme LILO. Ce dernier affiche alors le menu de tout ce qu'il connaît comme système. Lorsque l'utilisateur en sélectionne un en particulier, LILO va chercher dans ses tablettes à **quelle position physique du disque** se trouve son image kernel. Une fois cette position trouvée, il charge cette image en mémoire et l'exécute. Le kernel prend alors la main. Généralement il va charger ensuite un fichier `initrd` qui contient les pilotes de base pour être capable de comprendre ce qu'est un contrôleur IDE, un système de fichier EXT3, etc. Le kernel monte le système de fichier racine et est alors en mesure de poursuivre l'exécution jusqu'à la transmettre à `/sbin/init`. Là c'est le système GNU qui prends la main avec la détection des périphériques, chargement des services jusqu'à aboutir à l'invité de connexion.

GRUB vs LILO

Comme nous l'avons vu, LILO a besoin de connaître la position physique du kernel pour le charger en mémoire. Et ce kernel n'est jamais au même endroit selon le système de fichier utilisé (EXT3, Reiser, ZFS, etc.) et ce d'autant plus qu'une même partition peut contenir une floppée de kernels. Pour résoudre ce problème, il est nécessaire après chaque modification de la configuration de LILO de lancer la commande `lilo` qui va rechercher ces positions et les stocker en dur dans le MBR.

L'inconvénient majeur de cette approche est que si par exemple la géométrie du disque change, LILO est incapable de retrouver le kernel et plante au moment de le charger en mémoire. C'est un problème typiquement rencontré lorsque l'on transfère un disque dur d'une machine à une autre, avec deux contrôleurs IDE différents qui présente une géométrie du disque discordante à LILO.

Et c'est sur cet aspect que GRUB prend le relais surpasse son aîné car lui n'a aucun besoin de savoir à quelle position physique se trouve une image kernel, il la trouve tout seul comme un grand. En effet GRUB, contrairement à LILO sait lire un système de fichier, du moins la majorité d'entre eux. C'est pour cette raison que GRUB ne nécessite le lancement d'aucune commande de préparation après modification de sa configuration. Il fera tout le travail au démarrage.

Mais GRUB ne s'arrête pas là, il est aussi capable de rattraper une erreur de paramétrage au démarrage en permettant l'édition de la configuration à chaud ou en vous donnant la main sur un mini-interpréteur de commandes. Il peut aussi charger une image kernel à travers le réseau et ainsi démarrer un système sans disque dur.

Enfin, plus pour la petite histoire qu'autre chose, GRUB est un projet GNU dont la version 2 est déjà en route ^[1]. Ainsi donc, avec le remplacement quasi systématique de LILO par GRUB, Linux peut clairement être appelé GNU/Linux/GNU, au sens "boot sequence" du terme.

Traduction de la configuration

La première chose à faire est d'installer GRUB. Il est disponible en paquet sur toutes les distributions décentes, par exemple sous Mandriva par un `urpmi grub`.

Une fois le paquet installé, il nous faut transformer la configuration de LILO (`/etc/lilo.conf`) en menu GRUB (`/boot/grub/menu.lst`). Prenons en exemple un extrait du fichier `/etc/lilo.conf` :

```
image=/boot/vmlinuz
  label="linux"
  root=/dev/hda5
  initrd=/boot/initrd.img
  append="resume=/dev/hdc5"
```

```
vga=788
```

Ici nous demandons à LILO d'ajouter une entrée nommée `Linux` sur la partition `/dev/hda1` avec chargement d'un kernel se trouvant en `/boot/vmlinuz` (image compressée), et un fichier de pilotes en `/boot/initrd.img`. La variable `append` injecte des options supplémentaires au kernel (ici une partition d'hibernation) et enfin `VGA` ne fait qu'indiquer à LILO de passer en mode framebuffer 800x600 (788).

Nous allons donc commencer par créer un fichier `/boot/grub/menu.lst` commençant ainsi :

```
timeout 10
color black/cyan yellow/cyan
default 0
```

Cela indique seulement à GRUB que la première entrée est celle par défaut et qu'il faut attendre 10 secondes avant de l'amorcer si l'utilisateur ne touche à rien.

Ensuite vient la conversion de notre entrée LILO en langage GRUB :

```
title          Linux
root           (hd0,4)
kernel         /boot/vmlinuz root=/dev/hda5 resume=/dev/hdc5
initrd         /boot/initrd.img
vga            788
```

Pas très compliqué n'est-ce pas ? La seule chose, outre la syntaxe, qui change réellement avec LILO c'est la primitive `root (hd0,4)` qui va indiquer à GRUB de chercher l'image du kernel sur la partition n°5 du premier disque. Attention cependant, le numéro de la partition correspond à la numérotation donnée par Linux pour les `/dev/hdaX`. Ainsi si la partition est `/dev/hda7` ou `/dev/sda7`, le notation GRUB sera `(hd0,6)`.

Si vous aviez une partition Windows à démarrer, la syntaxe est un peu plus chaude car ne s'agissant pas d'un démarrage de kernel, il va falloir indiquer à GRUB comment procéder :

```
title WilainOS
root (hd0,0)
makeactive
chainloader +1
```

Cette syntaxe indique à GRUB de rendre la partition active, de charger le premier secteur et de démarrer là dessus.

Installation de GRUB dans le MBR

Comme GRUB est beaucoup plus gros que LILO, il ne tient pas entier dans les 512 octets de la MBR. Il est donc découpé en deux parties : `stage1` et `stage2`. Pour lancer leur installation un script est fourni en standard et s'utilise comme ceci :

```
root# /sbin/grub-install --recheck --no-floppy /dev/hda
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(hd0) /dev/hda
root#
```

Voilà, c'est terminé. Il ne vous reste plus qu'à redémarrer en ayant bien soin de garder à portée de main un CD-ROM bootable avec dessus un kernel de secours au cas où.

Maintenant si vous vous êtes planté par exemple dans les numéros de partition, "*don't panic !!*". C'est là que GRUB vous dévoile toute sa puissance. Il va couiner en disant que le système n'existe pas ou que la partition indiquée est introuvable et vous demander de presser une touche pour continuer. Ne vous faites pas prier vous aller ainsi revenir au menu. Ensuite pressez le touche `e` sur la ligne qui plante, cela vous affiche le paramétrage que vous avez rentré dans `menu.lst`. Sélectionnez la ligne qui contient l'erreur et pressez à nouveau `e` pour la modifier. Vous êtes en `qwerty` mais sinon pas de problème. Vous validez et lorsque le paramétrage vous semble bon, tapez `b` pour démarrer dessus, et ça roule. Magique non ? Ceci dit, une fois que le démarrage s'est correctement déroulé, allez corriger cela "pour de vrai" dans le fichier `menu.lst` pour éviter ce jonglage à chaque redémarrage.

Conclusion

Sa capacité à lire les systèmes de fichier et à permettre l'édition du paramétrage à chaud fait de GRUB une vraie bénédiction. Car techniquement l'un des immenses atout de Linux est son indépendance vis à vie du matériel. Une indépendance qui lui permettrait de démarrer dans à peu près n'importe quelles conditions s'il n'était pas bridé par ce très

tatillon de LILO. Avec GRUB, j'ai décroché un disque système avec apache, ldap, et tout le tintouin d'un carte mère ATX basée sur un AMD et l'ai coller sans ménagement sur une carte VIA C7 avec un CPU et un chipset radicalement différent. Et le tout a démarré tranquillement sans poser le moindre problème...

*<http://artisan.karma-lab.net/node/1605>
(C) artisan numerique - CC BY-SA*

Liens:

[1] <http://www.gnu.org/software/grub/grub-2.en.html>