

# Gestionnaire de démarrage Debian Sid

<http://www.linux-france.org/~platu/weblog/>

La révision système de mon transportable en cette période de rentrée m'a amené à redécouvrir les vertus du gestionnaire de démarrage LILO. Bien sûr, mes choix en matière de système de fichiers et de gestionnaire de volumes logiques (LVM2) peuvent paraître «singuliers». Si j'ai rencontré quelques difficultés, je ne peux m'en prendre qu'à moi-même ; surtout en utilisant la branche unstable de la distribution Debian. Voici un résumé des manipulations possibles pour récupérer un fonctionnement correct de gestionnaire de démarrage à partir d'une situation catastrophique : une machine inutilisable sur laquelle aucun système d'exploitation ne peut être lancé à partir du disque dur.

## Table des matières

1. Choix d'un gestionnaire de démarrage .....	1
2. Contexte Debian Sid .....	1
3. Knoppix 6.0 à la rescousse .....	2
4. Configuration LILO .....	3
5. Pour conclure .....	4

## 1. Choix d'un gestionnaire de démarrage

Les deux logiciels phares de gestion du lancement de système d'exploitation qui dominent le monde GNU/Linux sont : *LILO* et *GRUB*.

*LILO* ou *Linux LOader*<sup>1</sup> est la solution historique avec laquelle j'ai débuté il y a quelques années déjà. À l'usage, le fait de devoir passer par une écriture du *Master boot record*<sup>2</sup> (MBR) à chaque modification du ou des noyaux utilisables sur un système et le manque d'options lors de l'initialisation de la machine ont conduit à l'émergence d'un nouveau gestionnaire de démarrage. De plus, il semble que le développement de *LILO* ait été abandonné.

*GRUB* ou *GRand Unified Bootloader*<sup>3</sup> est la solution qui a gagné les faveurs de la majorité des distributions GNU/Linux. Relativement au gestionnaire historique, il est possible de modifier les paramètres de démarrage du système d'exploitation lors de l'initialisation de la machine. De plus, le système de menus permet d'offrir plus de choix aux utilisateurs novices.

Si l'on doit retenir une différence de conception importante entre les deux gestionnaires d'amorce, c'est la capacité à accéder à un système de fichiers. En effet, *LILO* utilise les fonctions du *BIOS*<sup>4</sup> pour accéder directement au contenu du disque dur sans se préoccuper du système de fichiers utilisé. À l'inverse, *GRUB* a besoin de reconnaître un système de fichiers *ext2*<sup>5</sup> pour accéder à l'ensemble de ses fonctions et menus.

Suivant les cas de figure, cet accès à un système de fichiers peut être un avantage ou un gros inconvénient comme on le voit ci-dessous.

## 2. Contexte Debian Sid

Même si les fonctionnalités offertes par *GRUB* sont très séduisantes, le développement de cet outil connaît quelques soubresauts assez gênants : Le billet *The grub drama*<sup>6</sup> publié sur *Planet Debian*<sup>7</sup> au printemps dernier résume assez bien les griefs nés entre utilisateurs et développeurs suite à la décision d'initier un nouveau développement.

Ensuite, les soucis récents sont venus des contradictions entre le développements des règles *udev*<sup>8</sup> de création des entrées de l'arborescence des périphériques et d'autres outils ; dont *GRUB*.

<sup>1</sup> [http://fr.wikipedia.org/wiki/Linux\\_loader](http://fr.wikipedia.org/wiki/Linux_loader)

<sup>2</sup> [http://fr.wikipedia.org/wiki/Master\\_boot\\_record](http://fr.wikipedia.org/wiki/Master_boot_record)

<sup>3</sup> [http://fr.wikipedia.org/wiki/GRand\\_Unified\\_Bootloader](http://fr.wikipedia.org/wiki/GRand_Unified_Bootloader)

<sup>4</sup> [http://fr.wikipedia.org/wiki/Basic\\_Input\\_Output\\_System](http://fr.wikipedia.org/wiki/Basic_Input_Output_System)

<sup>5</sup> <http://fr.wikipedia.org/wiki/Ext2>

<sup>6</sup> <http://blog.zugschlus.de/archives/826-The-grub-drama.html>

<sup>7</sup> <http://planet.debian.net/>

<sup>8</sup> <http://en.wikipedia.org/wiki/Udev>

L'histoire commence avec une énième mise à jour du noyau et un «pic de stress» en constatant que la liste des noyaux disponibles donnée par la commande **update-grub -v** est *vide*. La commande ci-dessus n'étant qu'un script *shell*, une nouvelle exécution en mode *debug* via la directive **sh -x** montre que c'est le programme **grub-probe** qui échoue lamentablement dans la reconnaissance du système de fichiers !

Une fois de plus, *Google* n'est pas mon ami et la quasi totalité des liens renvoyés correspondent à des listes de disques éronées dans le fichier `device.map`. Sur mon transporable, il n'y a qu'un disque SATA référencé `/dev/sda` ; donc peu de chance de se tromper de ce côté là.

Voici une copie du fichier `/etc/fstab` qui montre une utilisation du gestionnaire de volumes logiques LVM2.

```
$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/mapper/Amethyste-root / ext4 errors=remount-ro 0 1
/dev/sda1 /boot ext3 defaults 0 2
/dev/mapper/Amethyste-home /home ext4 defaults 0 2
/dev/mapper/Amethyste-tmp /tmp ext4 defaults 0 2
/dev/mapper/Amethyste-usr /usr ext4 defaults 0 2
/dev/mapper/Amethyste-var /var ext4 defaults 0 2
/dev/mapper/Amethyste-swap none swap sw 0 0
/dev/hda /media/cdrom0 udf,iso9660 user,noauto 0 0
```

Relativement à ces définitions de points de montages, l'exécution de la commande **mount** ne correspond pas tout à fait. Les entrées du répertoire `/dev/mapper` ne sont pas utilisées.

```
$ mount
/dev/dm-0 on / type ext4 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda1 on /boot type ext3 (rw)
/dev/dm-4 on /home type ext4 (rw)
/dev/dm-2 on /tmp type ext4 (rw)
/dev/dm-3 on /usr type ext4 (rw)
/dev/dm-5 on /var type ext4 (rw)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
capifs on /dev/capi type capifs (rw,mode=0666)
```

À ce moment là, je n'ai pas vu le lien entre les points de montages et **grub-probe**. J'ai donc tenté un redémarrage de la machine. C'est là que le drame s'est produit : plus de noyau accessible ! La machine était inutilisable.

### 3. Knoppix 6.0 à la rescousse

Dans mon [billet précédent sur les joies de la gestion de parc avec Ghost](#)<sup>9</sup>, j'avais déjà eu à utiliser la version 6.0.0 de **KNOPPIX**<sup>10</sup> pour jouer avec les propriétés du systèmes de fichiers *ext3*. Une fois de plus, ce *live-cd* a servi à récupérer une situation calamiteuse.

Une fois la machine initialisée avec **KNOPPIX**, je suis logiquement retombé sur le même problème avec **grub-probe**. Cette commande refusait obstinément de reconnaître le moindre système de fichiers. Il était bien possible d'installer **GRUB**, mais une fois de *Shell* lancé, les commandes **root DEVICE** et **setup DEVICE** échouaient.

C'est là que l'idée d'utiliser **LILO** s'est imposée. Il fallait nécessairement utiliser un gestionnaire d'amorce qui ne dépende absolument pas d'un accès aux systèmes de fichiers du disque de la machine.

<sup>9</sup> [http://www.linux-france.org/~platu/weblog/archives/2009/07/index.html#e2009-07-11T18\\_28\\_27.txt](http://www.linux-france.org/~platu/weblog/archives/2009/07/index.html#e2009-07-11T18_28_27.txt)

<sup>10</sup> <http://knoppix-fr.org/>

Là où l'utilisation de *KNOPPIX* devient intéressante, c'est que même lorsqu'un outil n'est pas disponible sur le CD il est toujours possible d'utiliser le gestionnaire de paquets et d'accéder via le réseau aux dépôts *Debian*. Une fois le câble réseau de la machine branché et une requête DHCP plus tard, il était possible d'installer *LILO*.

Lors de l'initialisation du CD, j'ai l'habitude de donner les options suivantes :

```
# knoppix lang=fr 2
```

Une fois que l'accès au réseau est établi, on utilise la gestion de paquets de façon tout à fait classique :

```
# apt-get update
# apt-get install lilo
```

## 4. Configuration LILO

La gestion de volume logiques n'est pas activée automatiquement lors de l'initialisation du *live-CD*. Il est donc nécessaire de le faire manuellement.

```
# modprobe dm-mod
# vgchange -a y
# lvdisplay -c
File descriptor 7 (/proc/6669/status) leaked on lvdisplay invocation. Parent PID 11242: bash
/dev/Amethyste/root:Amethyste:3:1:-1:1:1048576:128:-1:0:-1:254:0
/dev/Amethyste/swap:Amethyste:3:1:-1:2:4194304:512:-1:0:-1:254:1
/dev/Amethyste/tmp:Amethyste:3:1:-1:1:1048576:128:-1:0:-1:254:2
/dev/Amethyste/usr:Amethyste:3:1:-1:1:25165824:3072:-1:0:-1:254:3
/dev/Amethyste/home:Amethyste:3:1:-1:1:125829120:15360:-1:0:-1:254:4
/dev/Amethyste/var:Amethyste:3:1:-1:1:154787840:18895:-1:0:-1:254:5
```

Une fois que le noyau est au courant de la présence des volumes logiques, on peut commencer les manipulations sur le gestionnaire d'amorce. On commence par le montage manuel des partitions ou volumes nécessaires.

```
# mount /dev/Amethyste/root /mnt
# mount /dev/sda1 /mnt/boot
```

La partie la plus facile à traiter est l'écriture du *Master Boot Record*. Cette étape passée, *GRUB* à disparu.

```
# lilo -M /dev/sda mbr
```

Vient ensuite la phase de configuration à l'aide du fichier `/etc/lilo.conf`. Fort heureusement, un fichier exemple est fourni dans la documentation du paquet *LILO* : `/usr/share/doc/lilo/examples/conf.sample`. Sur la base de cet exemple, on s'adapte au contexte de la machine.

```
# cat /mnt/etc/lilo.conf
# /etc/lilo.conf: Sample Debian LILO boot loader configuration.

boot=/dev/sda1
root=/dev/mapper/Amethyste-root
compact
large-memory

# To use the new LILO boot menu, add the following
bitmap=/boot/sid.bmp
bmp-colors=1,,0,2,,0
bmp-table=120p,173p,1,15,17
bmp-timer=254p,432p,1,0,0
install=bmp

# install=menu
map=/boot/map
vga=normal
delay=20
timeout=150
prompt
```

```
image=/vmlinuz
root=/dev/mapper/Amethyste-root
initrd=/initrd.img
label=Linux
read-only

image=/vmlinuz
root=/dev/mapper/Amethyste-root
initrd=/initrd.img
label=Linux_single
append=" single "
read-only

image=/vmlinuz.old
root=/dev/mapper/Amethyste-root
initrd=/initrd.img.old
label=Linux_old
read-only
```

Dernière étape essentielle, il faut appliquer cette configuration à l'aide d'une nouvelle écriture dans le *Master Boot record*.

```
# lilo -r /mnt/
```

Après réinitialisation de la machine, on dispose à nouveau d'une liste de noyaux et le système d'exploitation se lance normalement. Une seule réaction OUF !

## 5. Pour conclure

Voilà comment échapper à une grosse galère sur la gestion du démarrage d'un système. Même si *LILO* n'est plus du tout le gestionnaire d'amorce en vogue, il peut s'avérer fort utile et le fait qu'il ne dépende d'aucun accès à un système de fichier est un avantage déterminant dans certains cas. Il est vraiment dommage que son développement se soit arrêté.

Dans le prochain billet, je dois traiter le problème de la gestion des entrées de périphériques *LVM2* via *udev* ou *dmsetup*. Tous mes soucis venaient en fait de là !

Ce document est disponible en version imprimable au format PDF : [lilo-revival.pdf](#)<sup>11</sup>.

\$Id: lilo-revival.xml 1417 2009-09-12 10:17:12Z latu \$

---

<sup>11</sup> <http://www.linux-france.org/~platu/weblog/telechargement/lilo-revival.pdf>