# Lilo mini-Howto

Cameron Spitzer (cls@truffula.sj.ca.us), Alessandro Rubini (rubini@linux.it).
Traduction française Mathieu Arnold *arn@mygale.org* v2.02, 16 August 1998

LILO est le **Li**nux **Lo**ader (chargeur de Linux) pour le version x86 de Linux. Je l'appellerai ici Lilo plutôt que LILO car je n'aime pas trop les majuscules. Ce fichier décrit quelques installations typiques de Lilo. Le but de ce document est d'être un supplément au guide de l'utilisateur de Lilo. Je pense que des exemples aident à comprendre, même si votre configuration n'est pas vraiment semblable à la mienne. J'espère que cela va vous éviter des tracas. Comme la documentation de Lilo est déjà très bonne, ceux qui sont intéressés par les détails iront faire un tour dans /usr/doc/lilo*

## Contents



# 1 Introduction

Bien que la documentation fournie avec les sources de Lilo (celle installée dans `/usr/doc/lilo`-version) soit très facilement compréhensible, la majorité des utilisateurs de Linux rencontrent des problèmes en faisant leur propre `/etc/lilo.conf`. Ce document a pour but d'aider les utilisateurs en leur donnant un minimum d'informations, et en leur montrant cinq exemples d'utilisation :

- Le premier exemple montre une installation de base : "Linux et les autres".
- La suivante explique comment installer Lilo sur un disque dur connecté sur `/dev/hdc` et qui bootera en tant que `/dev/hda`. C'est ce dont on a besoin quand on veux installer un nouveau disque Linux sur

un système qui marche déjà. Il explique aussi comment booter sur un disque SCSI quand votre BIOS est suffisamment récent.

- Le troisième exemple montre comment booter un système Linux dont la partition root ne peut être accédée par le BIOS.
- L'exemple suivant est utilisé pour accéder à de très gros disques auxquels ni le BIOS, ni le DOS ne peuvent accéder facilement (cet exemple ci est un peu vieux)
- Le dernier exemple montre comment restaurer un disque endommagé, si les dommages proviennent de l'installation d'un autre système d'exploitation.

Les trois derniers exemples sont de Cameron *cls@truffula.sj.ca.us* , qui est à l'origine de ce document. Alessandro *rubini@linux.it* , le mainteneur actuel n'a que Linux, et donc, ne peut pas vérifier ni non plus mettre à jour par lui même. Il n'est pas nécessaire de dire que toutes les remarques seront les bienvenues.

# 2 Informations générales et installation standard

Quand Lilo amorce le système, il utilise des appels du BIOS pour charger le noyau de Linux depuis le disque (disque IDE, disquette, ou autre). Par conséquent, le noyau doit résider à un endroit accessible par le BIOS.

Au boot, Lilo ne sais pas lire les données des systèmes de fichiers, et le path que vous avez mis dans `/etc/lilo.conf` est résolu à l'installation (quand vous avez lancé `/sbin/lilo`). L'installation est le moment où le programme construit les tables qui regroupent les secteurs qui sont utilisés pour charger le système d'exploitation. Par conséquent, tous les fichiers doivent être dans un endroit que le BIOS peut lire (les fichiers sont généralement placés dans le répertoire `/boot`, cela signifie que seule la partition root de votre système a besoin d'être accessible depuis le BIOS).

Une autre conséquence du fait de se baser sur le BIOS est que vous devez réinstaller le chargeur (en relançant */sbin/lilo*) à chaque fois que vous modifiez la configuration de Lilo. Lorsque vous recompilez le noyau et que vous remplacez votre image, vous devez réinstaller Lilo.

## 2.1 Où dois-je installer Lilo ?

La directive `boot=` de `/etc/lilo.conf` dit à Lilo où il doit placer son amorceur primaire. En général, vous spécifirez soit le Master Boot Record (MBR) (`/dev/hda`) soit la partition de root de Linux (par exemple `/dev/hda1` ou `/dev/hda2`).

Si vous avez un autre système d'exploitation installé sur votre disque, vous feriez mieux d'installer Lilo sur votre partition de boot plutôt que sur le MBR. Dans ce cas, vous devez rendre la partition “bootable” en utilisant la commande “a” de *fdisk* ou la commande “b” de *cfdisk*. Si vous n'écrivez pas sur le MBR, il sera plus simple de désinstaller Linux et Lilo si nécessaire.

## 2.2 Comment dois-je configurer mes disques IDE ?

Pour ma part je n'utilise pas les options LBA ou LARGE du BIOS (mais il faut dire que je n'utilise que Linux sur ma machine); ce sont des monstruosités qui nous sont imposées par les lacunes de l'architecture intel x86. Ceci implique que le noyau doit résider dans les 1024 premiers cylindres, mais cela n'est pas un problème vu que vous avez partitionné votre disque et que votre partition de root est sensée être petite (en tous cas, c'est comme ça que cela devrais être).

Si votre disque a un autre système d'exploitation, vous ne devez pas modifier les paramètres du BIOS, sinon, l'ancien système ne fonctionnera plus du tout. Toutes les distributions récentes de Lilo savent quoi faire des réglages LBA et LARGE.

Notez que le mot-clé "`linear`" dans le `/etc/lilo.conf` peut vous être utile si vous avez des problèmes de géométrie. Il indique à Lilo que l'adressage des secteurs doit se faire linéairement plutôt qu'avec le triplet secteurs/têtes/cylindres. La conversion des adresses 3D est reportée à l'exécution, par conséquent, rendant la configuration immunisée contre les problèmes de géométrie.

Si vous avez plusieurs disques et que certains qui ne sont utilisés que par Linux ne sont pas nécessaires durant le boot, vous pouvez les enlever de votre BIOS, votre système démarrera plus rapidement et Linux détectera tous les disques. Je change souvent les disques de mon ordinateur, mais je ne touche jamais au BIOS.

## 2.3 Comment puis-je interagir au boot ?

Quand vous voyez le prompt Lilo, vous pouvez taper sur la touche <Tab> pour voir les différents choix possible. Si Lilo n'est pas configuré pour être interactif, gardez la touche <Tab> ou <Shift> pressée avant que le message "LILO" n'apparaisse.

Si vous choisissez de booter sur un noyau Linux, vous pouvez ajouter des arguments après le nom du système que vous choisissez. Le noyau accepte de nombreux arguments. Tous les arguments sont listés dans le "BootPrompt-HOWTO" de Paul Gortmaker, et je ne vais pas le reproduire ici. Quelques arguments sont, par ailleurs, très importants et se doivent de figurer ici :

- `root=` : vous pouvez dire au noyau de monter une partition root différente de celle qui se trouve dans le `lilo.conf`. Par exemple, mon système a une toute petite partition qui a un système Linux minimal, et j'ai réussi à booter après avoir détruit ma partition root par erreur.
- `init=` : depuis la version 1.3.43 du noyau, vous pouvez utiliser une commande autre que `/sbin/init`. si vous avez de graves problèmes durant le démarrage, vous pouvez accéder à un système minimal en spécifiant `init=/bin/sh` (quand vous arriverez au prompt du shell, vous aurez certainement besoin de vos partitions, essayez "`mount -w -n -o remount /; mount -a`", et n'oubliez pas de faire un "`umount -a`" avant d'éteindre).
- Un nombre : en spécifiant un nombre sur la ligne de commande, vous demandez a *init* de démarrer dans un run-level spécifique (le run-level par défaut est généralement 3 ou 2 suivant la distribution que vous avez). Réferez-vous à la documentation d'*init*, à votre `/etc/inittab` et à vos `/etc/rc*.d` si vous voulez creuser plus loin.

## 2.4 Comment désinstaller Lilo ?

Quand Lilo écrit sur un secteur de boot, il en sauve une copie dans `/boot/boot.`*xxyy*, ou *xxyy* sont les nombres majeurs et mineurs du périphérique. Vous pouvez voir les nombres majeurs et mineurs de votre disque ou partition en lançant "`ls -l /dev/`*device*". Par exemple le premier secteur de `/dev/hda` (majeur 3, mineur 0) sera sauvé dans `/boot/boot.0300`, en installant Lilo sur `/dev/fd0` on aura un fichier `/boot/boot.0200` et sur `/dev/sdb3` créera `/boot/boot.0813`. Notez que Lilo ne créera pas le fichier s'il existe déjà, vous n'aurez donc pas à faire une sauvegarde quand vous réinstallerez Lilo (après avoir recompilé votre noyau par exemple). Les copies de `/boot/` sont toujours les sauvegardes d'avant la première installation de Lilo.

Si jamais vous avez besoin de désinstaller Lilo (par exemple, dans le cas où vous auriez malheureusement à désinstaller Linux), vous aurez besoin de restaurer le secteur de boot original. Si Lilo est installé sur `/dev/hda`, faites "`dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1`" (personnellement, je fais simplement "`cat /boot/boot.0300 > /dev/hda`", mais ce n'est pas toujours fiable, car cela restaurera la table des partitions, que vous avez peut être modifié depuis). Cette commande est bien plus simple que d'essayer de lancer "`fdisk /mbr`" depuis la ligne de commande DOS : elle vous permet de supprimer Linux d'un disque à partir de Linux. Après avoir supprimé Lilo, n'oubliez pas de supprimer la partition Linux avec *fdisk* (le *fdisk* du DOS est incapable de supprimer les partitions non-dos).

Si vous avez installé Lilo sur votre partition de root (par exemple `/dev/hda2`), vous n'avez rien de spécial à faire pour supprimer Lilo, lancez juste le *fdisk* de Linux pour supprimer la partition. Vous aurez aussi à marquer la partition DOS bootable.

# 3 La configuration simple

La majorité des installations Lilo utilisent un fichier de configuration tel celui ci :

```
boot = /dev/hda    # ou la partition root
delay = 10         # délai, en dixièmes de secondes
vga = 0            # optionnel, utilisez "vga=1" pour avoir du 80x50
#linear            # essayez ça si vous avez des problèmes de géométrie

image = /boot/vmlinux  # votre zImage
  root = /dev/hda1     # votre partition root
  label = Linux        # ou un nom rigolo
  read-only            # monter la root en lecture seule

other = /dev/hda4   # votre partition DOS, si y'en a une
  table = /dev/hda  # La table de partition courante
  label = dos       # ou un nom triste
```

Vous pouvez avoir plusieurs sections "images" et "other" si vous voulez. Il est très fréquent de rencontrer de multiples images de noyau dans votre *lilo.conf*, au moins si vous restez à jour dans le développement du noyau.

## 3.1 Comment s'en sortir avec les gros noyaux

Si vous compilez un noyau "zImage" et qu'il est trop gros pour rentrer dans un demi mégaoctet (ce qui est fréquent avec les noyaux 2.1), vous aurez à compiler un "big zImage" à la place : "`make bzImage`". Pour booter avec un gros noyau, vous n'aurez pas besoin de faire quoi que ce soit si votre version de Lilo est supérieure à 18. Si votre installation est plus ancienne, vous aurez à mettre à jour votre paquetage Lilo.

## 3.2 autres sources d'informations

En addition aux documentations de Lilo, il y a un très grand nombre de Mini-HowTo qui peuvent vous être utiles. Ils sont tous appelés "Linux+*nulOS*", pour quelques *nulOS*, ils parlent aussi de comment faire coexister Linux et des autres OS. De même, le "Multiboot-with-LILO" décrit comment les différentes versions de Windows coexistent avec Linux.

# 4 Configurons `hdc` pour booter en tant que `hda` en utilisant `bios=`

Lilo permet de mapper l'image du noyau sur un disque et de dire au BIOS d'aller la chercher sur un autre disque. Par exemple, j'ai l'habitude d'installer Linux sur un disque connecté sur `hdc` (disque maître du second contrôleur IDE) et d'y booter comme système unique sur le contrôleur primaire d'un autre ordinateur. J'ai copié la disquette d'installations sur une petite partition, je peut donc faire un *chroot* depuis une console virtuelle pour installer `hdc` quand j'utilise le système pour faire autre chose.

Voilà le `lilo.conf` que j'utilise pour installer Lilo :

```
# Ce fichier doit être utilisé sur un système fonctionnant à partir
# de /dev/hdc
boot = /dev/hdc   # on réécrit le MBR d'hdc
disk = /dev/hdc   # On lui dit qui sera hdc
   bios = 0x80    # Et le BIOS la verra en tant que premier disque
delay = 0
vga = 0

image = /boot/vmlinux  # c'est sur /dev/hdc1
  root = /dev/hda1     # Mais au boot, ça sera hda1
  label = Linux
  read-only
```

Ce fichier de configuration doit être lu par Lilo **depuis** `/dev/hdc1`. La table de Lilo qui est écrite sur le secteur de boot (`/dev/hdc`) doit se référer à un fichier dans `/boot/` (en ce moment `hdc`). Ce fichier sera accédé en tant que `hda` quand il bootera comme seul système.

J'ai appelé ce fichier `/mnt/etc/lilo.conf.hdc` (`/mnt` est l'endroit où hdc est monté durant l'installation. J'installe Lilo en lançant "`cd /mnt; chroot .  sbin/lilo -C /etc/lilo.conf.hdc`". Allez lire la page man de *chroot* si vous trouvez ça magique.

Le "`bios=`" du `lilo.conf` est utilisé pour dire à Lilo ce que le BIOS pense de vos périphériques. Le BIOS identifie les lecteurs de disquettes et les disques durs par des numéros : 0x00 et 0x01 sélectionnent les disquettes, 0x80 et suivants, les disques durs (Les vieux BIOS ne peuvent accéder qu'à deux disques). La signification du "`bios = 0x80`" dans l'exemple précédent est de dire à Lilo "utilise 0x80 dans tes appels BIOS pour `/dev/hdc`".

Cette directive Lilo peut être très pratique dans d'autres situations, par exemple, quand votre BIOS est capable de booter depuis un disque SCSI à la place d'un disque IDE. Quand des périphériques IDE et SCSI sont présents, LILO ne sait pas à qui appartient 0x80, car l'utilisateur est capable de le changer depuis les menus du BIOS, et le BIOS ne peut pas être accédé quand Linux est lancé.

Par défaut, Lilo suppose que les disques IDE sont mappés en premier par le BIOS, mais il est possible de lui spécifier le contraire en utilisant ces instructions dans `/etc/lilo.conf` :

```
disk = /dev/sda
  bios = 0x80
```

# 5 Utilisation de Lilo quand le BIOS ne voit pas la partition root

J'ai deux disques IDE, et un disque SCSI. Le disque SCSI ne peut pas être vu par le BIOS. Lilo utilise des appels BIOS, et peut uniquement voir ce que le BIOS voit. Mon stupide AMI BIOS ne peut booter que sur "A:" ou "C:", or ma partition de root se trouve sur le disque SCSI.

La solution consiste en fait à stocker le noyau, la carte, et la chaîne d'amorçage sur une partition Linux sur le premier disque IDE. Remarquez qu'il n'est pas nécessaire de garder le noyau sur la partition root.

La deuxième partition de mon premier disque IDE (`/dev/hda2`, la partition Linux utilisée pour booter sur le système) est montée sur `/u2`. Et voici le `/etc/lilo.conf` que j'utilise.

```
#  On installe Lilo sur le MBR du premier disque IDE
#
boot = /dev/hda
#  /sbin/lilo (l'installateur) copie le boot record de Lilo
#  depuis le fichier suivant vers le MBR
install = /u2/etc/lilo/boot.b
#
#  J'ai écrit un menu détaillé. Lilo le trouvera là
message = /u2/etc/lilo/message
#  L'installateur construira le fichier suivant. Il dit à
#  l'amorceur ou sont les blocs ou se trouvent les noyaux
map = /u2/etc/lilo/map
compact
prompt
#  on attende 10 seconds, puis on boot sur le noyau 1.2.1 par défaut.
timeout = 100
#  Le noyau est stocké ou le BIOS peut le trouver en faisant ça :
#      cp -p /usr/src/linux/arch/i386/boot/zImage /u2/z1.2.1
image = /u2/z1.2.1
        label = 1.2.1
#  Lilo dit au noyau de monter la première partition SCSI en tant que
#  root. Le BIOS n'a pas à savoir qu'elle existe.
        root = /dev/sda1
#  Cette partition sera vérifiée puis remontée par /etc/rc.d/rc.S
        read-only
#  J'ai un noyau d'une vielle Slackware qui traîne dans un coin au cas
#  ou j'ai un noyau qui ne marche pas. J'en ai déjà eu besoin une fois
image = /u2/z1.0.9
        label = 1.0.9
        root = /dev/sda1
        read-only
#  Ma partition DR-DOS 6
other = /dev/hda1
        loader=/u2/etc/lilo/chain.b
        label = dos
        alias = m
```

# 6 Accéder a de gros disques quand le BIOS n'en est pas capable

A mon travail, ma machine à un disque IDE de 1 Go. Le BIOS ne peut en voir que les premiers 504 Mo (où Mo signifie 2ˆ10 octets, pas 10ˆ6 octets). Donc, J'ai MS-DOS sur une partition de 350 Mo `/dev/hda1` et ma partition root Linux sur une partition de 120 Mo `/dev/hda2`.

MS-DOS était incapable de s'installer correctement quand le disque était tout neuf. Novell DOS 7 pareil. Le disque était supposé arriver avec un disque appelé “OnTrack Disk Manager”. Malheureusement pour moi, suite à un oubli de la part d'IBM je n'avais pas la disquette avec “OnTrack” avec le disque. Si vous n'avez que MS-DOS, je vous souhaite d'en disposer.

Donc, j'ai bâti une table de partition avec le *fdisk* de Linux. MS-DOS-6.2 a refusé de s'installer dans /dev/hda1, prétextant quelque chose du genre :

```
“Cette version de MS-DOS est dédiée aux nouvelles installations.  MS-DOS est déjà
installé sur votre ordinateur (ce qui était faux :  disque neuf) donc, vous avez besoin
d'obtenir une version de mise à jour chez votre vendeur.”
```

Quel ignare ! Donc, je relance le *fdisk* Linux et détruis la première partition de la table. Cela convient à MS-DOS 6.2 qui peut alors créer la même partition que celle que je viens de détruire et s'installer. MS-DOS 6.2 écrit alors dans le secteur de lancement du disque, mais impossible de démarrer.

Par chance, j'avais un noyau de la Slackware sur disquette (réalisé par le programme `setup` d'installation), et j'ai donc lancé Linux puis écrasé le secteur de démarrage par celui de Lilo... et tout marche !

Voici le fichier `/etc/lilo.conf` utilisé :

```
boot = /dev/hda
map = /lilo-map
delay = 100
ramdisk = 0             # Ne crée pas le disque virtuel du noyau Slackware
timeout = 100
prompt

disk = /dev/hda         # le BIOS ne voit que les 500 premiers Mo.
   bios = 0x80          # indique le premier IDE.
   sectors = 63         # prendre ces chiffres dans la documentation du disque
   heads = 16
   cylinders = 2100

image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
  read-only
  vga = extended

other = /dev/hda1
  label = msdos
  table = /dev/hda
  loader = /boot/chain.b
```

Après avoir installé ces systèmes, j'ai vérifié que les partitions contenant les fichiers zImage, boot.b, map, chain.b, et messages peuvent utiliser le système de fichiers MS-DOS, tant que ni Stacker, ni Doublespace ne sont installés. Donc, j'aurais pu faire une partition DOS sur /dev/hda1 de 500 Mo.

J'ai également appris que *OnTrack* aurait écrit une table de partitions commençant à quelques douzaines d'octets sur le disque au lieu de l'écrire au début. Il est possible de modifier le gestionnaire de périphérique IDE de Linux pour contourner ce problème. Mais l'installation aurait été impossible avec le noyau précompilé de la Slackware. En fin de compte, IBM m'a envoyé une disquette *OnTrack*. J'ai alors téléphoné au support technique de *OnTrack*. Ils m'ont dit que Linux était boggé car il n'utilisait pas le BIOS. J'ai renvoyé la disquette.

# 7 Démarrage depuis une disquette de secours

Ensuite, j'ai installé Windows-95 sur mon ordinateur au bureau. Il a détruit mon joli secteur de démarrage LILO, mais n'a pas touché à mes partitions Linux. Les noyaux sont très lents à charger à partir des lecteurs de disquettes. J'ai donc fait une disquette avec un fichier de configuration pour LILO qui me permet de lancer le noyau se trouvant sur le disque IDE. J'ai fait la disquette de cette manière :

```
fdformat /dev/fd0H1440  : formatage de la disquette vierge
```

```
mkfs /dev/fd0 1440      : disquette au format Minix
mkdir /3                : montage
mount /dev/fd0 /3
cp -p /boot/chain.b /3  : copie du chargeur
lilo -C /etc/lilo.flop  : installation de LILO
umount /3
```

Notez que la disquette DOIT ÊTRE MONTÉE LORSQUE VOUS LANCEZ L'INSTALLATION pour que LILO puisse écrire proprement le fichier de configuration.

Ce fichier est `/etc/lilo.flop`, il ressemble à celui-ci :

```
#  Crée une disquette qui puisse lancer des noyaux sur disque dur
boot = /dev/fd0
map = /3/lilo-map
delay = 100
ramdisk = 0
timeout = 100
prompt

disk = /dev/hda        # 1 Go IDE, BIOS ne voit que les premiers 500 Mo.
   bios=0x80
   sectors = 63
   heads = 16
   cylinders = 2100

image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
  read-only
  vga = extended

other = /dev/hda1
  label = msdos
  table = /dev/hda
  loader = /3/chain.b
```

Enfin, j'avais besoin de MS-DOS 6.2 sur mon ordinateur du bureau, mais je ne voulais pas toucher au premier disque. J'ai donc ajouté un contrôleur SCSI et un disque, créé une partition au format MS-DOS avec le programme `mkdosfs` de Linux, et Windows-95 l'a reconnu comme disque "D:". Mais, bien sûr, MS-DOS ne démarrera pas sur "D:". Ce n'est pas un problème avec LILO. J'ai ajouté :

```
other = /dev/sda1
  label = d6.2
  table = /dev/sda
  loader = /boot/any_d.b
```

au fichier `lilo.conf` de tout à l'heure. MS-DOS-6.2 fonctionne en pensant être sur C:, et Windows 95 est sur D:.