Guide pratique des disquettes d'amorçage

Version française du *Bootdisk HOWTO*

Tom Fawcett `<fawcett CHEZ croftj POINT net>`

Adaptation française: Mathieu Decore

`<mdecore CHEZ linux TIRET france POINT org>`

Adaptation française (version 3.5): Frank Pavageau

`<pavageau CHEZ imaginet POINT fr>`

Relecture de la version française: Guillaume Hatt

`<ghatt CHEZ netcourrier POINT com>`

Préparation de la publication: Jean-Philippe Guérard

`<jean TIRET philippe POINT guerard CHEZ laposte POINT net>`

Version 4.5.fr.1.0

20 juin 2003

Historique des versions

Version 4.5.fr.1.0	2003-06-20	MD, GH, JPG
Mise à jour de la version française. Conversion en XML (MD).		
Version 4.5	Janvier 2002	TF
Version 4.0	2000-04-01	TF
Version 3.5.fr.1.0	Juillet 1999	FP
Version 3.5	Juillet 1999	TF

Ce document explique comment concevoir et créer ses propres disquettes d'amorçage et disquettes racines pour Linux. Ces disquettes peuvent être utilisées comme disques de secours, ou pour l'essai de nouveaux composants système. Avant de tenter de créer une disquette d'amorçage, il est recommandé d'avoir un compréhension raisonnable de l'administration d'un système Linux. Si vous souhaitez juste créer des disquettes de secours, lisez la Section 1, « Disques d'amorce préfabriqués ».

Table des matières

1. Préface

Ce document peut être obsolète.

Si la date sur la page de titre a plus de six mois, vérifiez la page du Projet de documentation Linux (LDP) http://www.ibiblio.org/LDP/HOWTO/Bootdisk-HOWTO.html (et sa traduction française http://www.traduc.org/docs/howto/lecture/Bootdisk-HOWTO.html) au cas où une version plus récente s'y trouverait.

Bien que ce document soit lisible dans sa forme texte, il a *bien* meilleure allure en PostScript, PDF ou HTML en raison de la typographie utilisée.

1.1. Notes

Graham Chapman (`<grahamc CHEZ zeta POINT org POINT au>`) a écrit le premier Bootdisk-HOWTO et en a assuré la maintenance jusqu'à la version 3.1. Tom Fawcett (`<fawcett CHEZ croftj POINT net>`) a ajouté beaucoup d'informations pour le support du noyau 2.0, et effectue désormais la maintenance du document, depuis la version 3.2 qui contient toujours une bonne partie écrite par Chapman. Chapman a disparu de la communauté Linux et personne ne sait actuellement où il se trouve.

Les informations contenues dans ce document concernent Linux sur plate-forme Intel™. Bien des éléments doivent pouvoir être appliqués à Linux sur d'autres processeurs, mais nous n'en avons pas personnellement l'expérience, ni ne possédons d'informations de ce type. Si quelqu'un a expérimenté les disques d'amorce sur d'autres plates-formes, qu'il nous contacte.

1.2. À faire

1. User-mode-linux (http://user-mode-linux.sourceforge.net/) semble être un bon moyen de tester les disquettes d'amorce sans avoir à redémarrer la machine sans arrêt. Je ne l'ai pas testé. Si quelqu'un l'a fait fonctionner de manière significative pour fabriquer ses propres disquettes d'amorce, merci de me le faire savoir.

2. Ré-analyser la distribution de la disquettes d'amorce et mettre à jour la section « La méthode des pros »

3. Trouver jusqu'où la séquence init-getty-login peut être simplifiée, et la remplacer. Quelque personnes m'ont dit que **init** peut être un lien vers **/bin/sh** ; si c'est le cas, et que cela n'impose pas de grandes modifications, changer les instructions pour le faire. Cela permettrait d'éliminer l'utilisation de **getty**, **login**, **gettydefs**, et sans doute tout ce qui concerne PAM et NSS.

4. Aller à nouveau dans le code source du noyau 2.4 et écrire une explication détaillée sur la procédure de démarrage et de chargement du disque mémoire (seulement si je le comprends mieux). Il y a des fonctionnalités à propos de **initrd** et les limitations concernant les périphériques de démarrage (comme les cartes flash) que je ne comprends pas encore.

5. Supprimer la section qui décrit comment mettre à jour une disquettes d'amorce. C'est la plupart du temps source de trop de tracas.

6. Remplacer la commande **rdev** par les mots-clefs **LILO**.

1.3. Retours et remerciements

Tout retour, bon ou mauvais, sur le contenu de ce document est le bienvenu. Nous avons fait de notre mieux pour vérifier que les instructions et informations ici présentes sont précises et fiables. Merci de nous signaler toute erreur ou omission.

Nous remercions les nombreuses personnes nous ayant fourni des corrections et suggestions. Leurs contributions ont permis d'améliorer ce document bien au delà de ce que nous aurions pu réaliser seuls.

Envoyez vos commentaires, corrections et questions en anglais à l'auteur à l'adresse ci-dessus (ou au traducteur pour des problèmes dans la version française). Cela ne me dérange pas d'essayer de répondre à vos questions, mais merci de lire la Section 7, « En cas de problème, ou l'agonie de la défaite » d'abord.

1.4. Politique de distribution

Il est fortement conseillé d'effectuer une sauvegarde de votre système avant une installation importante ainsi qu'à intervalles réguliers.

1.5. Notes du traducteur

Correspondances anglais/français utilisées dans la traduction :

- I/O : E/S, entrées/sorties.
- backup : archive.
- boot loader : chargeur, lanceur.
- boot : amorce, amorcer, démarrage, démarrer.
- boot/root disk : disquette d'amorce/racine.
- dynamic library : bibliothèque partagée.
- filesystem : système de fichiers, système.
- inode : i-nœud.
- library : bibliothèque.
- loopback device : périphérique de boucle.
- package : paquetage.
- path : chemin.
- ramdisk : disque virtuel en mémoire, disque mémoire.
- root device : périphérique racine.
- root : racine.
- runlevel : niveau d'exécution.

- swap : pagination.

2. Introduction

Les disques d'amorce Linux sont utiles dans nombre de situations, telles que :

- Tester un nouveau noyau ;
- Redémarrer après un problème de disque : tout depuis la perte d'un secteur d'amorce à l'atterrissage d'une tête de lecture sur le disque ;
- Réparer un système endommagé. Une petite erreur en tant que root peut rendre le système inutilisable, et il peut alors être nécessaire de démarrer depuis une disquette pour corriger le problème ;
- Mettre à jour des fichiers critiques du système, tels que `libc.so`.

Les disques d'amorce peuvent être obtenus de différentes manières :

- Utiliser ceux d'une distribution telle que la Slackware. Cela vous permettra au moins de démarrer ;
- Utiliser un paquetage de création de disques de secours, prévus pour les cas d'urgence ;
- Apprendre ce que nécessite le fonctionnement de chaque type de disque, puis créer les vôtres.

Certaines personnes choisissent la dernière solution afin de tout faire eux-mêmes. Ainsi, si quelque chose ne marche plus, ils peuvent se débrouiller pour corriger le problème. Et c'est un excellent moyen pour apprendre le fonctionnement d'un système Linux.

Ce document suppose une certaine familiarité avec les concepts d'administration d'un système Linux. Par exemple, vous devez connaître les répertoires, les systèmes de fichiers, les disquettes. Vous devez savoir utiliser **mount** et **df**, à quoi servent les fichiers `/etc/passwd` et `fstab` et quelle tête ils ont. Enfin, vous devez savoir que la plupart des commandes dans ce *howto* doivent être exécutées en tant que root.

Créer vos propres disquettes d'amorce à partir de zéro peut être compliqué. Si vous n'avez pas lu la FAQ Linux et autres documents du même genre, tels que le *howto* d'installation Linux et le Guide d'installation de Linux, vous ne devriez pas essayer de créer des disques d'amorce. Si vous souhaitez juste créer des disques pour prévenir les cas urgents, il est *bien* plus simple d'en récupérer des préfabriqués. Lisez l'appendice Section 1, « Disques d'amorce préfabriqués » ci-dessous pour savoir où en trouver.

3. Disques d'amorce et démarrage

Un disque d'amorce est essentiellement un système Linux miniature et auto-suffisant contenu sur une disquette. Il doit pouvoir effectuer nombre d'opérations possibles sur un système Linux de taille normale. Avant d'essayer d'en créer un, vous devez comprendre les bases du démarrage de Linux. Nous en faisons ici une présentation qui devrait suffire à la compréhension de la suite du document. Bien des détails ou des possibilités ont été omis.

3.1. Démarrage

Tous les PC démarrent en exécutant du code situé en mémoire morte (à savoir, le BIOS) afin de charger le secteur situé au secteur 0, cylindre 0 du *disque d'amorce*. Celui-ci est habituellement le premier lecteur de disquette (appelé *A:* sous DOS et `/dev/fd0` sous Linux). Le BIOS essaye alors d'exécuter ce secteur. Sur la plupart des disques d'amorce, le secteur 0, cylindre 0 contient :

- soit le code d'un chargeur tel que **LILO**, qui trouve le noyau, le charge et l'exécute pour réaliser le démarrage proprement dit,
- soit le début du noyau d'un système d'exploitation, tel que Linux.

Si un noyau Linux a été copié directement sur une disquette, le premier secteur du disque sera le premier secteur du noyau Linux lui-même. Ce premier secteur continuera le démarrage en chargeant le reste du noyau depuis le périphérique d'amorce.

Une fois que le noyau est complètement chargé, il effectue certaines initialisations de périphériques ainsi que de ses données internes. Une fois qu'il est complètement initialisé, il consulte un endroit particulier dans son image appelé le *mot disque mémoire*. Ce mot précise comment et où trouver le système de fichiers racine. Un système de fichiers racine est simplement celui qui est monté en tant que « / ». Il faut dire au noyau où trouver ce système racine ; s'il ne peut trouver d'image à charger, il s'arrête.

Dans certains cas au démarrage (souvent lors du démarrage depuis une disquette), le système de fichiers racine est chargé dans un **disque mémoire**, auquel le système accède ensuite en mémoire comme s'il s'agissait d'un vrai disque. Il y a deux raisons à un tel chargement en mémoire. Premièrement, la mémoire vive est de plusieurs ordres de magnitude plus rapide qu'une disquette, et le système est donc rapide ; deuxièmement, le noyau peut charger un *système de fichiers compressé* depuis la disquette et le décompresser en mémoire, permettant ainsi de stocker plus de fichiers sur la disquette.

Une fois le système de fichiers racine chargé et monté, vous voyez un message tel que :

```
VFS: Mounted root (ext2 filesystem) readonly.
```

À ce moment, le système trouve le programme **init** dans le système racine (dans `/bin` ou `/sbin`) et l'exécute. **init** lit sa configuration dans `/etc/inittab`, cherche une ligne nommée *sysinit*, et exécute le script indiqué. Le script **sysinit** est en général un fichier du genre `/etc/rc` ou `/etc/init.d/boot`. Le script est un ensemble de commandes shell qui mettent en place les services de base du système, tels que :

- Lancer **fsck** sur tous les disques ;
- Charger les *modules* nécessaires du noyau ;
- Lancer la pagination ;
- Initialiser le réseau ;
- Monter les disques indiqués dans `fstab`.

Ce script appelle lui-même souvent nombre d'autres scripts pour réaliser une initialisation modulaire. Par exemple, dans la structure SysVinit habituelle, le répertoire `/etc/rc.d/` contient une structure complexe de sous-répertoires dont les fichiers indiquent comment lancer et arrêter la plupart des services du système. En revanche, sur un disque d'amorce, le script **sysinit** est souvent très simple.

Quand le script **sysinit** se termine, le contrôle revient à **init**, qui entre alors dans le *niveau d'exécution par défaut*, spécifié dans `inittab` par le mot clé *initdefault*. La ligne du niveau d'exécution indique en général un programme tel que **getty**, responsable de la gestion des communications par la console ou les `tty`. C'est le programme **getty** qui affiche l'invite habituelle *login:*. Lui-même exécute à son tour le programme **login** pour gérer la validation du login et mettre en place la session de l'utilisateur.

3.2. Types de disques

Après avoir revu les bases du démarrage, nous pouvons définir les différents types de disques mis en jeu. Ces disques sont classés en quatre catégories. Ça et là dans le document, on emploie le terme *disque* pour désigner une disquette, sauf précision contraire, bien que la majeure partie de la discussion puisse également s'appliquer à des disques durs.

Amorce
: Un disque contenant un noyau pouvant être amorcé. Il peut être utilisé pour démarrer le noyau, qui pourra alors charger un système racine depuis un autre disque. Il est en général nécessaire d'indiquer au noyau où se trouve ce système racine.

  Souvent un disque d'amorce charge le système racine depuis une autre disquette, mais il est possible de lui indiquer un disque dur d'où il chargera le système racine à la place. Souvent, c'est dans le but de tester un nouveau noyau (d'ailleurs, **make zdisk** crée un tel disque d'amorce automatiquement à partir du code source du noyau).

Racine
: Un disque avec un système de fichiers contenant les fichiers nécessaires au fonctionnement d'un système Linux. Un tel disque ne contient pas forcément un noyau ou un chargeur.

  Un disque racine peut être utilisé pour faire tourner le système indépendamment de tout autre disque, une fois le noyau amorcé. En général le disque racine est copié automatiquement vers un disque mémoire. Cela permet un accès au disque racine bien plus rapide, et libère le lecteur pour une disquette d'utilitaires.

Amorce/racine
: Un disque contenant à la fois un noyau et un système de fichiers racine. Autrement dit, il contient tout ce qui est nécessaire au démarrage et au fonctionnement d'un système Linux sans disque dur. L'avantage de ce type de disque est sa compacité : tout ce dont on a besoin est sur un seul disque. Néanmoins, la taille toujours plus importante des programmes implique une difficulté croissante pour tout faire tenir sur une seule disquette, même avec de la compression.

Utilitaire
: Un disque contenant un système de fichier non destiné à être monté en tant que racine. Il s'agit d'un disque de données supplémentaires. Vous pouvez utiliser ce genre de disque pour rajouter des utilitaires, quand vous en avez trop pour un seul disque amorce.

En général, lorsque l'on parle de « construire un disque d'amorce », c'est de la création des parties amorce (noyau) et racine (fichiers) qu'il est question, soit en un seul morceau (un seul disque amorce/racine), soit séparément (un disque amorce et un racine). L'approche la plus flexible pour des disquettes de secours est d'utiliser des disquettes amorce et racine séparées, et une ou plusieurs disquettes utilitaires pour supporter le trop-plein.

4. Construire un système racine

Pour créer un système racine, il faut sélectionner les fichiers nécessaires au système pour fonctionner. Dans cette section nous décrivons comment créer un *système racine compressé*. Une option moins courante est de créer un système non compressé sur une disquette que l'on monte directement ; cette alternative est décrite dans la Section 9.1, « Système racine sans disque mémoire ».

4.1. Aperçu

Un système racine doit contenir tout ce qui est nécessaire au bon fonctionnement d'un système Linux complet. Pour cela, le disque doit contenir un système Linux minimum :

- La structure de base des fichiers ;
- Un ensemble minimum de répertoires : `/dev`, `/proc`, `/bin`, `/etc`, `/lib`, `/usr`, `/tmp` ;
- Un ensemble d'utilitaires de base : **sh**, **ls**, **cp**, **mv**, et cætera ;
- Un ensemble minimum de fichiers de configuration : **rc**, `inittab`, `fstab`, et cætera ;
- Des périphériques : `/dev/hd*`, `/dev/tty*`, `/dev/fd0`, et cætera ;
- Des bibliothèques d'exécution fournissant les fonctions de base nécessaires aux utilitaires.

Bien sûr, n'importe quel système devient utile dès que l'on peut faire tourner quelque chose dessus, et une disquette racine ne devient en général utilisable que lorsque vous pouvez faire quelque chose du genre :

- Contrôler un système de fichiers sur un autre disque ; par exemple, pour contrôler le système racine de votre disque dur, vous devez pouvoir démarrer Linux depuis un autre disque, telle qu'une disquette racine. Vous pouvez alors lancer **fsck** sur votre disque racine habituel tant qu'il n'est pas monté ;
- Récupérer tout ou partie de votre disque racine initial à partir d'une sauvegarde en utilisant des utilitaires d'archivage et de compression tels que **cpio**, **tar**, **gzip** et **ftape**.

Nous décrirons comment construire un système *compressé*, ainsi appelé car il est compressé sur disque et qu'une fois démarré, le noyau le décompresse dans un disque mémoire. Avec un système compressé vous pouvez faire tenir beaucoup de fichiers (à peu près six méga-octets) sur une disquette standard de 1440 ko. Puisque le système de fichiers est bien plus gros que la disquette, il ne peut être construit directement sur la disquette. Il nous faut le construire ailleurs et le compresser avant de le copier sur la disquette.

4.2. Création du système de fichiers

Pour créer un tel système racine, il vous faut un autre périphérique capable de stocker tous les fichiers avant leur compression. Ce périphérique doit pouvoir contenir à peu près quatre méga-octets. Plusieurs solutions s'offrent à vous :

- Utiliser un *disque mémoire* (PÉRIPHÉRIQUE = `/dev/ram0`). Dans ce cas, la mémoire est utilisée pour simuler un disque physique. Le disque mémoire doit être suffisamment grand pour contenir un système de fichiers de la bonne taille. Si vous utilisez **LILO**, cherchez dans votre fichier de configuration (`/etc/lilo.conf`) une ligne du type :

  ```
  RAMDISK_SIZE = nnn
  ```

  qui détermine combien de mémoire peut être au plus allouée à un disque mémoire. La valeur par défaut est de 4096 ko, ce qui devrait suffire. Il ne sert probablement à rien de créer un tel disque mémoire sur une machine possédant moins de 8 Mo de RAM.

  Vérifiez que vous avez un périphérique tel que `/dev/ram0`, `/dev/ram` ou `/dev/ramdisk`. Si ce n'est pas le cas, créez `/dev/ram0` avec mknod (numéro majeur 1, numéro mineur 0).
- Une partition de disque dur inutilisée et assez grande (plusieurs méga-octets) est aussi une bonne solution.
- Utiliser un *périphérique de boucle (loopback)*, qui permet d'utiliser un fichier comme s'il s'agissait d'un périphérique normal. Avec un périphérique de boucle, vous pouvez créer un fichier de 3 méga-octets sur votre disque dur et construire le système de fichiers dedans.

Tapez **man losetup** pour savoir comment utiliser un périphérique de boucle. Si vous n'avez pas **losetup**, vous pouvez le récupérer, ainsi que des versions compatibles de **mount** et **umount**, dans le paquetage *util-linux* disponible dans le répertoire ftp://ftp.win.tue.nl/pub/linux-local/utils/util-linux/ [ftp://ftp.win.tue.nl/pub/linux-local/utils/util-linux/].

Si vous n'avez pas de périphérique de boucle (`/dev/loop0`, `/dev/loop1`, et cætera) sur votre système, vous devez en créer un avec **mknod /dev/loop0 b 7 0**. Une fois les exécutables **mount** et **umount** spéciaux installés, créez un fichier temporaire sur le disque dur suffisamment grand (par exemple, `/tmp/fsfile`). Vous pouvez utiliser une commande du type :

```
dd if=/dev/zero of=/tmp/fsfile bs=1k count=nnn
```

pour créer un fichier de *nnn* blocs.

Utilisez le nom du fichier à la place de PÉRIPHÉRIQUE ci-dessous. Quand vous exécutez une commande **mount**, vous devez inclure l'option `-o loop` pour dire au programme d'utiliser un périphérique de boucle. Par exemple :

```
mount -o loop -t ext2 /tmp/fsfile /mnt
```

va monter `/tmp/fsfile` (par périphérique de boucle) sur le point de montage `/mnt`. Un coup de **df** le confirmera.

Après avoir choisi une de ces options, préparez le PÉRIPHÉRIQUE avec :

```
dd if=/dev/zero of=PÉRIPHÉRIQUE bs=1k count=4096
```

Cette commande initialise à zéro le périphérique. Cette étape est importante pour la compression ultérieure du système de fichiers, afin que toutes les portions inutilisées soient remplies de zéros pour une compression maximum. Gardez cela à l'esprit si vous déplacez ou effacez des fichiers sur le système de fichiers. Le système de fichiers va désallouer correctement les blocs, *mais ne va pas les initialiser à zéro à nouveau* . Si vous effectuez beaucoup d'effacements et de copies, votre système de fichiers compressé risque d'être bien plus grand que nécessaire.

Ensuite, créez le système de fichiers. Le noyau Linux sait charger automatiquement deux types de système de fichiers dans un disque mémoire : minix et ext2, avec une préférence pour ext2. Si vous le choisissez, vous voudrez peut-être utiliser l'option `-i` afin de créer plus d'i-nœuds que par défaut ; `-i 2000` est une bonne valeur qui vous évitera de tomber à court d'i-nœuds. Vous pouvez sinon économiser des i-nœuds en supprimant pas mal de fichiers `/dev/` inutiles. **mke2fs** crée par défaut 360 i-nœuds sur une disquette de 1,44 Mo. Je trouve 120 i-nœuds largement suffisants pour ma disquette racine de secours, mais si vous conservez tous les périphériques dans le répertoire `/dev` vous dépasserez facilement les 360. L'utilisation d'un système racine compressé permet de créer un système de fichiers plus grand, contenant donc plus d'i-nœuds par défaut, mais vous pourrez quand même vouloir réduire le nombre de fichiers ou augmenter le nombre d'i-nœuds.

Vous allez donc taper une commande du genre :

```
mke2fs -m 0 -i 2000 PÉRIPHÉRIQUE
```

(Si vous utilisez un périphérique de boucle, le fichier que vous utilisez doit être indiqué à la place de ce *`PÉRIPHÉRIQUE`*. **mke2fs** vous demandera alors si vous voulez vraiment faire cela ; répondez oui.)

La commande **mke2fs** détectera automatiquement l'espace disponible et se configurera selon ce dernier. Le paramètre `-m 0` permet de ne pas réserver d'espace pour root, et laisse donc plus d'espace disponible sur le disque.

Ensuite, montez le périphérique :

```
mount -t ext2 PÉRIPHÉRIQUE /mnt
```

(Vous devez créer un point de montage `/mnt` s'il n'existe pas encore.) Dans les sections suivantes, tous les répertoires destination sont supposés relatifs à `/mnt`.

4.3. Remplissage du système de fichiers

Voici un minimum raisonnable de répertoires à créer sur votre système racine 1 :

- `/dev` : Périphériques, nécessaires aux E/S ;
- `/proc` : Répertoire de base nécessaire au système de fichiers proc ;
- `/etc` : Fichiers de configuration du système ;
- `/sbin` : Exécutables systèmes critiques ;
- `/bin` : Exécutables de base considérés comme partie intégrante du système ;
- `/lib` : Bibliothèques partagées nécessaires à l'exécution des programmes ;
- `/mnt` : Un point de montage pour la maintenance des autres disques ;
- `/usr` : Utilitaires et applications supplémentaires.

Trois de ces répertoires resteront vides sur les systèmes racine, il suffit donc de les créer avec **mkdir**. Le répertoire `/proc` n'est qu'une base sous laquelle le système proc est placé. `/mnt` et `/usr` ne sont que des points de montage utilisés une fois que le système amorce/racine tourne. Encore une fois, il suffit de créer ces répertoires.

Les quatre autres répertoires sont décrits dans les sections suivantes.

4.3.1. `/dev`

Tous les systèmes Linux ont besoin d'un répertoire `/dev` contenant un fichier spécial par périphérique accessible au système. Le répertoire en lui-même est normal, et peut être créé avec **mkdir** de la manière habituelle. Les fichiers spéciaux de périphérique doivent par contre être créés différemment, à l'aide de la commande **mknod**.

Il y a un raccourci par contre : copiez le contenu de votre répertoire `/dev` existant, puis supprimez ceux dont vous n'avez pas besoin. Il suffit juste de copier les fichiers spéciaux avec l'option `-R`. Cela copie le répertoire sans tenter de copier le contenu des fichiers. *Attention à bien utiliser un R en majuscule !* Si vous utilisez l'option en minuscule `-r`, vous allez vous retrouver en train de copier le contenu complet de votre disque dur — ou au moins tout ce que pourra en contenir une disquette ! Prenez donc vos précautions, et utilisez par exemple les commandes :

```
cp -dpR /dev/fd[01]* /mnt/dev
cp -dpR /dev/tty[0-6] /mnt/dev
```

en supposant que la disquette est montée sur `/mnt`. Les options `dp` demandent la copie des liens symboliques en tant que lien, plutôt que celle du fichier qui se trouve au bout de celui-ci, et la

1 La structure de répertoires présentée ici concerne une disquette racine seule. Les vrais systèmes Linux obéissent à un ensemble de règles bien plus complexes et contrôlées, appelé le Filesystem Hierarchy Standard [http://www.pathname.com/fhs/2.2/], pour déterminer où les fichiers doivent aller.

conservation des attributs originaux des fichiers, pour garder les bons propriétaires.

Si vous voulez le faire vous-mêmes, utilisez **ls -l** pour afficher les numéros majeurs et mineurs des périphériques qui vous intéressent, et créez-les sur la disquette en utilisant **mknod**.

Quelle que soit la manière retenue pour copier les périphériques, il faut vérifier que tous les périphériques dont vous aurez besoin sont bien présents sur la disquette de secours. Par exemple, **ftape** utilise les périphériques de bande, qu'il vous faudra donc tous copier si vous comptez utiliser votre lecteur de bande depuis le disque amorce.

À noter qu'un i-nœud est nécessaire pour chaque fichier de périphérique, et que les *i-nœuds* sont parfois une ressource rare, spécialement sur les systèmes de fichiers sur disquette. Il n'est donc pas idiot d'enlever du répertoire `/dev` de la disquette tous les fichiers de périphérique dont vous n'avez pas besoin. Bien des périphériques ne sont clairement pas nécessaires sur des systèmes spécifiques. Par exemple, si vous n'avez pas de disques SCSI vous pouvez tranquillement enlever tous les fichiers commençant par `sd`. De même, si vous ne comptez pas utiliser de port série vous pouvez supprimer tous les fichiers commençant par `ttyS`.

Si, en construisant le système de fichiers, vous obtenez l'erreur :

```
No space left on device
```

et que la commande **df** indique qu'il reste de l'espace disponible, c'est sans doutes qu'il n'y a plus d'i-nœud disponible. Un **df -i** affichera l'utilisation des i-nœuds.

N'oubliez pas d'inclure les fichiers suivants dans le répertoire : `console`, `kmem`, `mem`, `null`, `ram0`, `tty1`.

4.3.2. /etc

Ce répertoire doit contenir un certain nombre de fichiers de configuration. Ce qu'il devrait contenir dépend des programmes que vous avez l'intention d'exécuter. Sur la plupart des systèmes, on peut les répartir en trois groupes :

1. Nécessaires à tout moment, par exemple `rc`, `fstab`, `passwd` ;
2. Peut-être nécessaires, mais on n'en est pas sûr ;
3. Du bazar oublié là.

Les fichiers non essentiels peuvent être identifiés avec la commande :

```
ls -ltru
```

Les fichiers sont classés dans l'ordre inverse de dernière date d'accès, donc tout fichier qui n'est jamais lu peut être exclu d'une disquette racine.

Sur mes disquettes racine, je n'ai que 15 fichiers de configuration. Mon travail se réduit alors à gérer trois groupes de fichiers :

1. Ceux que je dois configurer pour un système d'amorce et racine :
   a. `rc.d/*` : scripts de démarrage du système et de changement de niveau d'exécution ;
   b. `fstab` : liste des systèmes de fichiers à monter ;
   c. `inittab` : paramètres pour le processus **init**, le premier à être lancé au démarrage.

d. `gettydefs` : paramètres pour le processus **init**, le premier à être lancé au démarrage.

2. Ceux que je dois nettoyer pour un système d'amorce et racine :

   a. `passwd` : liste des utilisateurs, des répertoires utilisateurs, et cætera ;

   b. `group` : groupes d'utilisateurs ;

   c. `shadow` : mots de passe cachés des utilisateurs. Il se peut que vous n'ayez pas ce fichier ;

   d. `termcap` : la base de données de fonctionnalités des terminaux.

   Si la sécurité est importante, `passwd` et `shadow` doivent être nettoyés pour ne pas copier de mots de passe d'utilisateurs hors du système et pour qu'en cas de démarrage sur disquette, les logins indésirables soient rejetés.

   Assurez-vous que `passwd` contienne au moins *root*. Si vous comptez donner accès à d'autres utilisateurs, vérifiez l'existence de leurs répertoires utilisateurs et de leurs shells.

   `termcap`, la base de données de terminaux, fait en général plusieurs centaines de kilo-octets. Vous devrez faire du ménage dans la version de votre disquette d'amorce/racine pour ne conserver que le ou les terminaux que vous utilisez, ce qui se réduit en général à l'entrée *linux* ou *linux-console*.

3. Le reste. Ils fonctionnent très bien tels quel, je ne les modifie donc pas.

Parmi tout cela, je n'ai en réalité que deux fichiers à configurer, et ils ne doivent contenir qu'étonnamment peu de choses.

- `rc` doit contenir :

  ```
  #!/bin/sh
  /bin/mount -av
  /bin/hostname Kangaroo
  ```

  Vérifiez qu'il est exécutable, qu'il contient bien une ligne **#!/bin/sh** au début et que ce sont les bons répertoires. Il n'est pas réellement nécessaire de lancer **hostname**, mais cela donne juste une meilleure allure.

- `fstab` doit au moins contenir :

  ```
  /dev/ram0       /       ext2    defaults
  /dev/fd0        /       ext2    defaults
  /proc           /proc   proc    defaults
  ```

  Vous pouvez copier des lignes de votre vrai `fstab`, mais vous ne devriez pas monter automatiquement de partitions de votre disque dur ; utilisez le mot clé *noauto* pour celles-là. Votre disque peut être endommagé ou mort quand vous utilisez le disque d'amorce.

Votre `inittab` doit être modifié pour que la ligne **sysinit** lance **rc** ou quelque autre script basique d'amorce. De plus, si vous ne souhaitez pas que les utilisateurs se loguent sur les ports série, commentez toutes les entrées **getty** qui font référence à des périphériques `ttys` ou `ttyS` à la fin de la ligne. Laissez les ports `tty` pour pouvoir vous connecter sur la console.

Un fichier `inittab` minimal contient ce qui suit :

```
id:2:initdefault
si::sysinit:/etc/rc
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
```

Le fichier `inittab` décrit ce que va lancer le système dans divers états, dont le démarrage, le passage en mode multi utilisateurs, et cætera. Attention aux noms de fichiers référencés dans `inittab` ; si **init** ne peut trouver le programme, le disque d'amorce s'arrêtera, et vous n'aurez peut-être même pas de message d'erreur.

Notez que certains programmes ne peuvent être déplacés en raison d'autres programmes qui référencent en dur leur position. Par exemple sur mon système, `/etc/shutdown` référence en dur `/etc/reboot`. Si je déplace **reboot** vers **/bin/reboot**, et que je lance une commande **shutdown**, elle va échouer en ne trouvant pas le fichier **reboot**.

Pour le reste, copiez juste tous les fichiers texte de votre répertoire `/etc`, ainsi que tous les exécutables présents dans `/etc` dont vous n'êtes pas sûr de pouvoir vous passer. Basez-vous sur l'exemple de l'Annexe C, *Exemple de contenu de répertoires sur un disque racine*. Il vous suffira probablement de copier ces fichiers, mais les systèmes pouvant être très différents, il n'est pas certain que le même ensemble de fichiers sur votre système soit équivalent aux fichiers listés. La seule méthode sûre est de partir d'`inittab` et d'en déduire ce qui est nécessaire.

La plupart des systèmes utilisent maintenant un répertoire `/etc/rc.d/` contenant des scripts shell pour les différents niveaux d'exécution. Il faut au minimum avoir un script **rc** unique, mais il peut être plus simple de carrément copier `inittab` et le répertoire `/etc/rc.d` depuis votre système puis de nettoyer les scripts shell dans le répertoire `rc.d` pour enlever tous les traitements inutiles pour un système sur disquette.

4.3.3. `/bin` et `/sbin`

Le répertoire `/bin` est un endroit pratique pour tous les utilitaires nécessaires aux opérations de base, tels que **ls**, **mv**, **cat** et **dd**. Voir l'Annexe C, *Exemple de contenu de répertoires sur un disque racine* pour un exemple d'ensemble de fichiers pouvant aller dans les répertoires `/bin` et `/sbin`. Il ne contient aucun des utilitaires nécessaires à la récupération d'une sauvegarde, tels que **cpio**, **tar** et **gzip**. C'est parce que je place ceux-ci sur une disquette utilitaire séparée, pour conserver de la place sur la disquette d'amorce et racine. Une fois la disquette d'amorce/racine démarrée, elle est copiée sur le disque mémoire, laissant ainsi le lecteur de disquette libre pour en monter une autre, la disquette utilitaire. En général je la monte sur `/usr`.

La création d'une *disquette utilitaire* est décrite ci-dessous dans la Section 9.2, « Construire un disque utilitaire » . Il est probablement souhaitable d'y maintenir une copie des mêmes versions d'utilitaires de sauvegarde que ceux utilisés pour écrire les sauvegardes, histoire de ne pas perdre de temps en essayant d'installer des versions qui ne peuvent pas lire vos bandes de sauvegarde.

Vérifiez que vous y mettez les programmes suivants : **init**, **getty** ou un équivalent, **login**, **mount**, un shell capable de faire tourner votre script **rc**, un lien de **sh** vers le shell en question.

4.3.4. `/lib`

Vous mettez dans `/lib` les bibliothèques partagées et les chargeurs nécessaires. Si les bibliothèques nécessaires ne sont pas trouvées dans `/lib`, le système ne pourra pas démarrer. Avec de la chance, un message vous expliquera pourquoi.

Pratiquement tous les programmes ont au moins besoin de la bibliothèque `libc`, `libc.so.`*N*, *N* étant le numéro de version courant. Vérifiez votre répertoire `/lib`, `libc.so.N` est en général un lien symbolique vers un fichier avec un numéro de version complet :

```
% ls -l /lib/libc.so*
-rwxr-xr-x   1 root     root      4016683 Apr 16 18:48 libc-2.1.1.so*
lrwxrwxrwx   1 root     root           13 Apr 10 12:25 libc.so.6 -> libc-2.1.1.so*
```

Dans le cas présent, il vous faut `libc-2.1.1.so`. Pour trouver les autres bibliothèques nécessaires, il faut lancer la commande **ldd** sur tous les exécutables que vous prévoyez de mettre sur la disquette. Par exemple :

```
% ldd /sbin/mke2fs
libext2fs.so.2 => /lib/libext2fs.so.2 (0x40014000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0x40026000)
libuuid.so.1 => /lib/libuuid.so.1 (0x40028000)
libc.so.6 => /lib/libc.so.6 (0x4002c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Tous les fichiers à droite sont nécessaires. Le fichier peut en réalité être un lien symbolique.

Notez que certaines bibliothèques sont *assez grosses* et ne tiendront pas facilement sur votre système racine. Par exemple, la `libc.so` citée précédemment fait environ 4 méga-octets. Vous devrez probablement nettoyer les bibliothèques avant de les copier sur votre système racine. Reportez-vous à la Section 8, « Réduire la taille du système racine » pour plus d'informations.

Il faut également inclure dans `/lib` un chargeur pour les bibliothèques. Il s'agira soit de `ld.so` (pour les bibliothèques a.out), soit de `ld-linux.so` (pour les bibliothèques ELF). Les versions récentes de **ldd** vous indiquent de quel chargeur vous avez besoin, comme dans l'exemple ci-dessus, mais de plus anciennes versions ne le font pas forcément. Si vous ne savez pas duquel vous avez besoin, utilisez la commande **file** sur la bibliothèque. Par exemple :

```
% file/lib/libc.so.4.7.2 /lib/libc.so.5.4.33 /lib/libc-2.1.1.so
/lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC), stripped
/lib/libc.so.5.4.33: ELF 32-bit LSB shared object, Intel 80386, version 1, stripped
/lib/libc-2.1.1.so: ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
```

Le mot *QMAGIC* indique que *4.7.2* est pour les bibliothèques a.out, et *ELF* que *5.4.33* et *2.1.1* sont pour les ELF.

Copiez le ou les chargeurs dont vous avez besoin sur le système racine que vous êtes en train de construire. Les bibliothèques et chargeurs doivent être testés *attentivement* avec les exécutables inclus. Si le noyau ne peut charger une bibliothèque nécessaire, il s'arrêtera en général brutalement, sans message d'erreur.

4.4. Utilisation de PAM et NSS

Votre système peut utiliser des bibliothèques chargées dynamiquement mais invisibles pour `ldd`. Si vous ne les incluez pas, vous risquez de ne pas pouvoir vous connecter ou utiliser votre disquette d'amorce.

4.4.1. PAM (Pluggable Authentication Modules)

Si votre système utilise PAM (Pluggable Authentication Modules, soit Modules Externes d'authentification), tenez-en compte dans la construction de votre disque d'amorce, sans quoi vous ne pourrez pas vous connecter. En quelques mots, PAM est une méthode modulaire sophistiquée pour authentifier les utilisateurs et contrôler leur accès aux services. Pour déterminer simplement si votre système utilise PAM, cherchez dans le répertoire `/etc` de votre disque dur un fichier `pam.conf` ou un répertoire `pam.d` ; si l'un des deux existe, vous devez prévoir un minimum de support pour PAM. (Vous pouvez aussi lancer `ldd` sur votre exécutable `login` ; si la sortie contient `libpam.so`, vous avez besoin de PAM.)

Heureusement, la sécurité est rarement un problème avec les disques d'amorce étant donné que quiconque avec un accès physique à la machine peut en général faire tout ce qu'il veut dessus. Vous pouvez donc complètement désactiver PAM en créant un fichier `/etc/pam.conf` simple sur votre système racine contenant :

```
OTHER   auth       optional     /lib/security/pam_permit.so
OTHER   account    optional     /lib/security/pam_permit.so
OTHER   password   optional     /lib/security/pam_permit.so
OTHER   session    optional     /lib/security/pam_permit.so
```

Copiez également le fichier `/lib/security/pam_permit.so` sur votre système racine. Cette bibliothèque ne fait qu'environ 8 ko et ne coûte donc pas grand chose.

Notez bien que cette configuration donne à tous un accès complet aux fichiers et services de votre machine. Si vous avez des impératifs de sécurité sur votre disque d'amorce pour une raison ou une autre, vous devrez copier une partie, voire l'ensemble de la configuration PAM de votre disque dur vers le système racine. Lisez bien attentivement la documentation de PAM, et copiez toutes les bibliothèques nécessaires depuis `/lib/security` vers votre système racine.

Vous devez aussi inclure `/lib/libpam.so` sur le disque racine. Mais vous le saviez déjà puisque vous avez lancé **ldd** sur `/bin/login` qui vous a montré cette dépendance.

4.4.2. NSS (Name Service Switch)

Si vous utilisez *glibc* (appelée aussi *libc6*), vous devez tenir compte des services de noms sans quoi vous ne pourrez pas vous connecter. Le fichier `/etc/nsswitch.conf` contrôle les recherches dans les bases de données pour divers services. Si vous ne comptez pas accéder à des services du réseau (tels que des recherches DNS ou NIS), un simple fichier `nsswitch.conf` comme suit suffit :

```
passwd:     files
shadow:     files
group:      files
hosts:      files
services:   files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
automount:  files
aliases:    files
netgroup:   files
publickey:  files
```

Ce fichier spécifie que tous les services ne sont fournis que par des fichiers locaux de la machine. Vous devez aussi inclure le fichier `/lib/libnss_files.so.X`, où X vaut 1 pour une glibc 2.0 et 2 pour une glibc 2.1. Cette bibliothèque est chargée dynamiquement pour gérer les recherches dans les fichiers.

Si vous comptez accéder au réseau depuis votre disque d'amorce, vous pouvez créer un fichier `nsswitch.conf` plus complet. Voir la page de manuel de `nsswitch` pour plus de détails. N'oubliez pas d'inclure un fichier `/lib/libnss_`*service*`.so.1` pour chaque *service* que vous ajoutez.

4.5. Modules

Si votre noyau est modulaire, vous devez savoir quels modules vous voudrez charger depuis votre disque d'amorce une fois le système démarré. Il vous faudra inclure les modules **ftape** et **zftape** si vos sauvegardes sont sur bandes, les modules pour périphériques SCSI si vous en avez, et éventuellement ceux pour le support PPP ou SLIP si vous souhaitez accéder au réseau en cas d'urgence.

Ces modules doivent être placés dans `/lib/modules`. Vous devez aussi inclure **insmod**, **rmmod** et **lsmod**. Si vous souhaitez charger les modules automatiquement, prenez **modprobe**, **depmod** et **swapout**. Et si vous utilisez **kerneld**, prenez le avec son fichier de configuration `/etc/conf.modules`.

Néanmoins, le principal avantage d'utiliser les modules est que vous pouvez déplacer les modules non essentiels sur un disque utilitaire et ne les charger que lorsque c'est nécessaire, ce qui prend alors moins de place sur le disque racine. Si vous devez gérer beaucoup de périphériques, il vaut mieux procéder de cette manière plutôt que de construire un seul gros noyau contenant tous les gestionnaires.

Attention, pour démarrer avec un système de fichiers ext2 compressé, vous devez avoir inclus le support pour disque mémoire et ext2. Ils ne peuvent être installés par des modules.

4.6. Quelques ultimes détails

Certains programmes, tels que **login**, se plaignent si le fichier `/var/run/utmp` et le répertoire `/var/log` n'existent pas.

Donc :

```
mkdir -p /mnt/var/{log,run}
touch /mnt/var/run/utmp
```

Enfin, après avoir installé toutes les bibliothèques dont vous avez besoin, lancez **ldconfig** pour refabriquer `/etc/ld.so.cache` sur le système racine. Le cache indique au loader où trouver les bibliothèques. Pour refabriquer `ld.so.cache`, lancez la commande suivante :

```
ldconfig -r /mnt
```

4.7. C'est dans la poche

Une fois le système racine construit, démontez-le, copiez-le dans un fichier et compressez-le :

```
umount /mnt
dd if=PÉRIPHÉRIQUE bs=1k | gzip -v9 > rootfs.gz
```

Une fois cette étape effectuée, vous obtenez un fichier `rootfs.gz` contenant votre système racine compressé. Vérifiez sa taille pour être sûr qu'il tient sur une disquette. Si ça n'est pas le cas vous devrez y retourner pour supprimer certains fichiers. La Section 8, « Réduire la taille du système racine » vous donnera des astuces pour y arriver.

5. Choisir un noyau

Vous avez maintenant un système de fichiers racine complet et compressé. La prochaine étape consiste à construire ou choisir un noyau. Dans la plupart des cas, vous pouvez copier votre noyau courant et démarrer la disquette avec. Cependant dans certains cas, vous voudrez peut-être construire un noyau différent.

La taille peut jouer. Si vous faites une disquette d'amorce/racine unique, le noyau va être un des plus gros fichiers de la disquette et il vaut donc mieux essayer d'en réduire la taille au maximum. Pour ce faire, construisez-le avec le minimum de fonctionnalités nécessaires au bon fonctionnement du système cible. Cela implique de retirer tout ce dont vous n'avez pas besoin. Le support réseau est un bon candidat, tout comme le support pour tout type de disque et de périphérique dont vous n'avez pas l'usage sur un système lancé par disquette. Comme indiqué précédemment, votre noyau doit contenir le support pour disque mémoire et ext2.

Une fois les fonctionnalités nécessaires déterminées, vous devez trouver ce qu'il faut rajouter.

L'utilisation la plus courante d'une disquette d'amorce/racine est l'examen et la récupération d'un système racine endommagé, ce qui peut nécessiter le support de certaines fonctionnalités supplémentaires dans le noyau. Par exemple, si vos sauvegardes sont stockées sur bande avec **ftape** pour gérer l'accès au lecteur de bande et que vous perdez votre disque racine ainsi que ceux contenant **ftape**, vous ne pourrez plus récupérer vos sauvegardes depuis les bandes. Vous devrez réinstaller Linux, rapatrier et réinstaller **ftape**, puis essayer de lire vos sauvegardes.

Ce que je veux dire, c'est que quelle que soit la méthode d'E/S que vous utilisez au niveau du noyau pour les sauvegardes, elle doit se trouver aussi sur votre noyau d'amorce.

La procédure de construction d'un noyau est décrite dans la documentation fournie avec celui-ci. C'est assez simple à suivre, vous pouvez donc commencer par faire un tour dans `/usr/src/linux`. Si vous avez des problèmes pour construire un noyau, vous ne devriez probablement pas essayer de faire de disquette d'amorce/racine de toute manière. Pensez à compresser le noyau avec **make zImage**.

6. Assemblage et fabrication de la ou des disquettes

Vous avez maintenant un noyau et un système de fichiers compressé. Si vous construisez un disque d'amorce/racine unique, vérifiez que leur taille ne dépasse pas celle du disque. Si vous avez un découpage sur deux disquettes, vérifiez que la taille du système racine ne dépasse pas celle de la disquette.

Il vous faut choisir entre l'utilisation de **LILO** pour démarrer le noyau du disque d'amorce et la copie du noyau directement sur la disquette d'amorce, sans **LILO**. L'avantage de **LILO** est la possibilité de passer des paramètres au noyau, ce qui peut être nécessaire pour initialiser votre matériel (Regardez le fichier `/etc/lilo.conf` sur votre machine. S'il existe et contient une ligne du type *append*=..., vous avez besoin de passer des paramètres). Son inconvénient est une complexité accrue dans la construction du disque d'amorce, ainsi qu'une place occupée sur la disquette légèrement plus importante. Vous devrez configurer un petit système de fichier séparé que nous appellerons le *système noyau*, où vous transférerez le noyau ainsi que quelques autres fichiers nécessaires à **LILO**.

Si vous décidez d'utiliser **LILO**, continuez la lecture ; si par contre vous voulez copier le noyau directement sur la disquette, passez directement à la Section 6.2, « Transfert du noyau sans LILO ».

6.1. Transfert du noyau avec LILO

La première chose à faire est de vérifier que vous avez une version récente de **LILO**. Ensuite, il faut créer un petit fichier de configuration pour **LILO**. Il doit ressembler à :

```
boot      =/dev/fd0
install   =/boot/boot.b
map       =/boot/map
read-write
backup    =/dev/null
compact
image     = NOYAU
label     = Bootdisk
root      =/dev/fd0
```

Pour la signification de ces paramètres, voir la documentation utilisateur de **LILO**. Il vous faudra probablement aussi rajouter une ligne *append*=... à ce fichier, comme dans le fichier `/etc/lilo.conf` de votre disque dur.

Sauvez-le en tant que `bdlilo.conf`.

Vous devez maintenant créer un petit système de fichier, que nous appellerons *système noyau*, pour le différencier du système racine.

Tout d'abord, calculez la taille que celui-ci doit faire. Prenez la taille de votre noyau en blocs (la taille donnée par **ls -s NOYAU** et ajoutez 50. Cinquante blocs sont en gros la taille nécessaire aux i-nœuds ainsi qu'aux autres fichiers. Vous pouvez calculer le nombre exact si vous voulez, ou simplement utiliser 50. Si vous créez un ensemble avec deux disques, vous pouvez carrément surestimer l'espace nécessaire puisque le disque n'est utilisé que par le noyau de toute manière. Appelez ce nombre *BLOCS_NOYAU*.

Mettez une disquette dans le lecteur (pour simplifier, supposons qu'il s'agit de `/dev/fd0`) et créez le système noyau ext2 dessus :

```
mke2fs -N 24 -m 0 /dev/fd0 BLOCS_NOYAU
```

L'option `-N 24` indique que l'on souhaite 24 i-nœuds, ce qui est largement suffisant pour ce système de fichiers. Ensuite, montez le système, supprimez le répertoire `lost+found` et créez des répertoire `dev` et `boot` pour **LILO** :

```
mount /dev/fd0 /mnt
rm -rf /mnt/lost+found
mkdir /mnt/{boot,dev}
```

Ensuite, créez les périphériques `/dev/null` et `/dev/fd0`. Au lieu de chercher leurs numéros de périphériques, vous pouvez simplement les copier depuis votre disque dur avec l'option `-R` :

```
cp -R /dev/{null,fd0} /mnt/dev
```

LILO a besoin d'une copie de son chargeur d'amorce, `boot.b`, que vous pouvez trouver sur votre disque dur. Il est d'habitude dans le répertoire `/boot`.

```
cp /boot/boot.b /mnt/boot
```

Enfin, copiez le fichier de configuration de **LILO** que vous avez créé précédemment avec votre noyau. Les deux peuvent être placés dans le répertoire racine :

```
cp bdlilo.conf NOYAU /mnt
```

Tout ce dont **LILO** a besoin est maintenant sur le système noyau, vous pouvez donc le lancer. Le paramètre `-r` de **LILO** est utilisé pour installer le chargeur sur une autre racine que la courante :

```
lilo -v -C bdlilo.conf -r /mnt
```

LILO doit s'exécuter sans erreur, après quoi le système noyau devrait ressembler à :

```
total 361
  1 -rw-r--r--   1 root     root          176 Jan 10 07:22 bdlilo.conf
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:23 boot/
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:22 dev/
358 -rw-r--r--   1 root     root       362707 Jan 10 07:23 vmlinuz
boot:
total 8
  4 -rw-r--r--   1 root     root         3708 Jan 10 07:22 boot.b
  4 -rw-------   1 root     root         3584 Jan 10 07:23 map
dev:
total 0
```

```
  0 brw-r-----   1 root     root        2,   0 Jan 10 07:22 fd0
  0 crw-r--r--   1 root     root        1,   3 Jan 10 07:22 null
```

Ne vous inquiétez pas si la taille des fichiers n'est pas exactement la même que la votre.

Laissez maintenant le disque dans le lecteur et allez à la Section 6.3, « Mise en place du mot disque mémoire ».

6.2. Transfert du noyau sans LILO

Si vous n'utilisez *pas* **LILO**, transférez le noyau sur le disque d'amorce avec la commande **dd** :

```
% dd if=NOYAU of=/dev/fd0 bs=1k
353+1 records in
353+1 records out
```

Dans ce exemple, **dd** a écrit 353 enregistrements complets, plus 1 partiel, ce qui signifie que le noyau occupe les 354 premiers blocs de la disquette. Appelez ce nombre *BLOCS_NOYAU* et pensez à l'utiliser dans la section suivante.

Enfin, indiquez que le périphérique racine doit être la disquette elle-même, et que le noyau doit être chargé en lecture/écriture.

```
        rdev /dev/fd0 /dev/fd0
        rdev -R /dev/fd0 0
```

Attention à bien utiliser un `-R` majuscule dans la seconde commande **rdev**.

6.3. Mise en place du mot disque mémoire

Le *mot disque mémoire* situé dans l'image du noyau permet de spécifier où se trouve le système racine, ainsi que d'autres options. Le mot peut être lu et modifié avec la commande **rdev**, et sa valeur s'interprète de la manière suivante :

```
bits  0-10 :    Décalage jusqu'au début du disque mémoire, en blocs
                de 1024 octets
bits 11-13 :    Inutilisé
bit     14 :    Drapeau indiquant s'il faut charger un disque mémoire
bit     15 :    Drapeau pour faire une pause avant de charger le
                système racine
```

Si le bit 15 est mis à 1, le noyau vous demandera au moment du démarrage de changer la disquette dans le lecteur. C'est nécessaire si vous utilisez un ensemble de deux disques. Il y a deux cas, suivant que vous créez une disquette d'amorce/racine unique ou un ensemble « amorce+racine » séparé.

1. Si vous créez un disque unique, le système racine compressé sera placé juste après le noyau, et donc le décalage sera le premier bloc libre (qui doit être au même endroit que *BLOCS_NOYAU*). Le bit 14 sera mis à 1, et le bit 15 à 0.

   Supposons par exemple que vous construisiez un disque unique dont le système racine doit commencer au bloc 253 (valeur décimale). Le mot disque mémoire devrait valoir 253 (toujours en décimal) avec le bit 14 à 1 et le bit 15 à 0. Pour calculer sa valeur vous pouvez simplement additionner les valeurs décimales. 253 + (2^14) = 253 + 16384 = 16637. Si vous ne comprenez pas d'où sort ce nombre, entrez-le dans une calculatrice scientifique et convertissez-le en binaire.

2. Si vous créez par contre un ensemble de deux disques, le système racine sera au bloc zéro du second disque, et le décalage sera donc zéro. Le bit 14 sera mis à 1 tout comme le bit 15. La valeur décimale sera donc de 2^14 + 2^15 = 49152 dans ce cas.

Après avoir bien calculé la valeur du mot disque mémoire, écrivez-le avec **rdev -r**. Attention à utiliser la valeur *décimale*. Si vous utilisez **LILO**, l'argument de **rdev** doit être le *chemin d'accès au noyau monté*, c'est à dire `/mnt/vmlinuz` ; si vous avez copié le noyau avec **dd**, utilisez à la place le nom du périphérique du lecteur de disquette (c'est à dire `/dev/fd0`).

```
rdev -r NOYAU_OU_LECTEUR_DE_DISQUETTE VALEUR
```

Si vous avez utilisé **LILO**, démontez maintenant la disquette.

Ne croyez pas ce que dit la page de manuel de **rdev/ramsize** à propos de la taille du disque mémoire. Cette page est obsolète. À partir du noyau 2.0, le mot disque mémoire ne détermine plus la taille du disque mémoire ; au lieu de cela, ce mot est déterminé à partir du tableau donné au début de cette section. Pour une explication détaillée, voir le fichier de documentation `ramdisk.txt` ou http://www.tamacom.com/tour/linux/S/9075.html [http://www.tamacom.com/tour/linux/S/9075.html].

6.4. Transfert du système racine

La dernière étape concerne le transfert du système racine.

- Si le système racine doit être placé sur le même disque que le noyau, transférez-le avec **dd** et son option `seek`, qui indique combien de blocs il faut sauter :

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k seek=BLOCS_NOYAU
  ```

- Si le système racine doit être placé sur un second disque, sortez la première disquette, mettez la seconde dans le disque, puis transférez-y le système racine :

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k
  ```

Bravo, vous avez fini ! *Vous devriez toujours tester un disque d'amorce avant de le ranger jusqu'à la prochaine urgence !* S'il n'arrive pas à démarrer, continuez à lire.

7. En cas de problème, ou l'agonie de la défaite

Lorsque l'on crée des disques d'amorce, les premiers essais n'amorcent souvent pas la machine. En général, la méthode utilisée consiste à construire le disque racine à partir de composants de votre système actuel pour essayer d'obtenir que le système de la disquette commence à afficher des messages sur la console. Une fois qu'il a commencé à vous parler, la bataille est presque gagnée puisque vous pouvez voir de quoi il se plaint et corriger les problèmes un à un jusqu'à ce que le système fonctionne normalement. Si le système s'arrête brutalement sans explication, il peut être difficile d'en trouver la cause. Pour que le système en arrive au point où il commence à afficher ses messages, un certain nombre de composants doivent être présents et bien configurés. La procédure à suivre pour déterminer les raisons du silence de votre système est la suivante :

- Si vous voyez un message du genre :

```
Kernel panic: VFS: Unable to mount root fs on XX:YY
```

C'est un problème courant qui ne peut avoir que quelques causes. Tout d'abord, cherchez le périphérique *XX:YY* dans la liste de codes des périphériques dans le fichier `/usr/src/linux/Documentation/devices.txt` ; s'agit-il du bon périphérique racine ? Si ce n'est pas le bon, vous n'avez sans doute pas lancé **rdev -R**, ou alors sur la mauvaise image. Si le code du périphérique est correct, vérifiez attentivement quels gestionnaires de périphériques ont été compilés dans le noyau. Assurez-vous que le support pour le lecteur de disquettes, les disques mémoires et le système de fichiers ext2 sont bien inclus ;

- Si vous voyez plusieurs erreurs du genre :

```
end_request: I/O error, dev 01:00 (ramdisk), sector NNN
```

Il s'agit d'une erreur d'E/S du périphérique disque mémoire, la plupart du temps à cause du noyau qui tente d'écrire au-delà de la fin du périphérique. Le disque mémoire est trop petit pour supporter la totalité du système de fichiers. Vérifiez les message d'initialisation du noyau pour trouver une ligne du type :

```
Ramdisk driver initialized : 16 ramdisks of 4096K size
```

Vérifiez la taille du système de fichiers *compressé*. Si le disque mémoire n'est pas assez grand, il faut l'agrandir.

- Vérifiez que le disque racine contient bien les répertoires que vous croyez. Il est facile de se tromper de niveau d'arborescence et de se retrouver avec quelque chose du genre `/racine/bin` au lieu de `/bin` sur votre disquette racine ;
- Vérifiez qu'il y a un `/lib/libc.so` avec le même lien que celui présent dans le répertoire `/lib` de votre disque dur ;
- Vérifiez que tous les liens symboliques du répertoire `/dev` de votre système actuel existent également sur le système racine, quand ces liens pointent vers des périphériques inclus sur la disquette racine. Notamment, les liens vers `/dev/console` sont souvent essentiels ;
- Vérifiez que vous avez inclus les fichiers `/dev/tty1`, `/dev/null`, `/dev/zero`, `/dev/mem`, `/dev/ram` et `/dev/kmem` ;
- Vérifiez la configuration de votre noyau : le support pour toutes les ressources nécessaires jusqu'à l'invite de connexion doit être directement inclus et non pas sous forme de modules. *Le support des disques mémoires et de ext2 doivent donc être présents dans le noyau* ;
- Vérifiez que le périphérique racine et le disque mémoire sont correctement configurés dans le noyau.

Une fois ces points généraux vérifiés, vous pouvez vous pencher sur ces points plus précis :

1. Vérifiez qu'**init** est présent en tant que `/sbin/init` ou `/bin/init`, et qu'il est exécutable ;
2. Lancez **ldd init** pour vérifier les bibliothèques d'**init**. Il n'y a normalement que `libc.so`, mais sait-on jamais… Vérifiez que vous avez bien inclus les bibliothèques et leurs chargeurs ;
3. Vérifiez que vous avez le bon chargeur pour vos bibliothèques : `ld.so` pour a.out et `ld-linux.so` pour ELF ;
4. Vérifiez le contenu de `/etc/inittab` sur la disquette d'amorce et ses éventuels appels à **get-**

ty (ou tout autre programme du genre, tel que **agetty**, **mgetty** ou **getty_ps**).

Comparez-le plusieurs fois avec l'`inittab` de votre disque dur. Vérifiez les pages de manuel du programme que vous utilisez pour être sûr de sa cohérence. `inittab` peut être le morceau le plus difficile en raison de sa syntaxe et du contenu qui dépendent de la version d'**init** utilisée et de la nature du système. La seule manière de s'en débarrasser, c'est de lire les pages de manuel d'**init** et `inittab` afin de comprendre exactement ce que fait le système lorsqu'il démarre. Vérifiez que `/etc/inittab` contient bien une entrée concernant l'initialisation du système. Elle doit contenir une commande lançant le script d'initialisation du système, qui doit lui aussi exister ;

5. Comme pour **init**, lancez **ldd** sur votre **getty** pour voir ses besoins, et vérifiez que les bibliothèques et chargeurs nécessaires sont présents sur le système racine ;

6. Assurez-vous d'avoir inclus un exécutable de shell (par exemple **bash** ou **ash**) capable de faire tourner tous vos scripts **rc** ;

7. Si vous avez un fichier `/etc/ld.so.cache` sur le disque de secours, refabriquez-le (le fichier, pas le disque).

Si **init** démarre et que vous obtenez un message du type :

```
Id xxx respawning too fast: disabled for 5 minutes
```

cela provient d'**init** et indique généralement que **getty** ou **login** meurt aussitôt après son lancement.

Vérifiez les exécutables de **getty** et **login**, et les bibliothèques dont ils dépendent. Vérifiez que les appels depuis `/etc/inittab` sont corrects. Si vous obtenez d'étranges messages de **getty**, cela peut signifier que les arguments dans `/etc/inittab` sont faux. Les options des programmes **getty** sont variables ; on signale que les arguments sont parfois incompatibles entre deux versions d'**agetty**.

Si vous obtenez une invite de login et qu'après avoir entré un nom de login valide, le système vous en demande un autre aussitôt, le problème peut venir de PAM ou NSS. Lisez la Section 4.4, « Utilisation de PAM et NSS ». Le problème peut aussi venir du fait que vous utilisez les *mots de passe cachés* et que vous n'avez pas copié le fichier `/etc/shadow` sur votre disque d'amorce.

Si vous essayez de lancer un exécutable tel que **df** présent sur votre disque de secours, mais n'obtenez qu'un message du type : *df: not found*, vérifiez deux chose : (1) que le répertoire contenant le binaire est bien dans votre PATH, et (2) que vous avez les bibliothèques (et chargeurs) nécessaires au programme.

8. Réduire la taille du système racine

Parfois un système racine est trop gros pour tenir sur une disquette, même après compression. Voici quelques techniques pour réduire sa taille, citées par ordre décroissant d'efficacité :

8.1. Augmentez la densité du disque

Par défaut, les disquettes sont formatées à 1440 ko, mais des formats plus denses existent. `fdformat` peut formater des disques avec les tailles suivantes : 1600, 1680, 1722, 1743, 1760, 1840 et 1920. Lisez la page de manuel de `fdformat` ainsi que `/usr/src/linux/Documentation/devices.txt`.

Mais quelles densités/géométries votre machine peut-elle supporter ? Voici des réponses (légèrement modifiées) de Alain Knaff, l'auteur de **fdutils**.

> C'est plus le problème du BIOS que du format physique des disquettes. Si le BIOS décide que tous les numéros de secteurs supérieurs à 18 sont non valides, alors on

> ne peut pas y faire grand chose. En effet, à moins de désassembler le BIOS, le seul moyen de trouver la bonne valeur est en tâtonnant. Quoiqu'il en soit, si le BIOS supporte les disques ED (grande densité : 36 secteurs/piste et 2,88 Mo), il y a des chances pour que les disquettes de 1722 ko soient également supportées.
>
> Les disquettes super-formatées avec plus de 21 secteurs par piste ne sont vraisemblablement pas amorçables : en fait, celles qui utilisent des secteurs de tailles non standard (1024 octets par secteur au lieu de 512, par exemple) ne sont vraisemblablement pas démarrables. Il devrait être malgré tout possible d'écrire un programme de démarrage du secteur pour contourner cela. Si je me souviens bien, le programme **DOS 2m** en est capable, ainsi que les programmes **XDF** de OS/2.
>
> Certains BIOS clament artificiellement que tout secteur supérieur à 18 est certainement défectueux. Comme les disquettes de 1722 ko utilisent des secteurs supérieurs à 21, elles ne devraient pas être amorçables. Le meilleur moyen de tester serait de formater une disquette DOS ou syslinux en 1722 ko et de la rendre amorçable. Si vous utilisez **LILO**, n'utilisez pas l'option `linear` (sans quoi **LILO** penserait que le format par défaut de la disquette est de 18 secteurs par piste, et la disquette ne démarrera pas même si le BIOS la supporte).
>
> —Alain Knaff

8.2. Remplacer les utilitaires indispensables par BusyBox

La plupart de l'espace disque d'un système de fichiers est consommé par des utilitaires indispensables tels que **cat**, **chmod**, **cp**, **dd**, **df**, et cætera. Le projet *BusyBox* permet de fournir un remplacement à ces utilitaires indispensables. BusyBox fournit un seul fichier monolithique exécutable, **/bin/busybox**, d'environ 150 ko, qui implémente les fonctions de ces utilitaires. Vous pouvez créer des liens symboliques à partir de différents programmes vers cet exécutable ; **busybox** voit comment il a été appelé et invoque le code correcte. BusyBox inclut même un shell basique. BusyBox est disponible sous forme de paquetage binaire pour plusieurs distributions, et le code source est disponible sur le site de BusyBox.

8.3. Changez de shell

Certains shells populaires sous Linux, tels que **bash** et **tcsh**, sont gros et nécessitent de nombreuses bibliothèques. Si vous n'utilisez pas le shell de BusyBox, vous devriez quand même songer à remplacer le shell. D'autres options plus légères existent, telles que **ash**, **lsh**, **kiss** et **smash**, bien plus petites et nécessitant peu (ou pas) de bibliothèques. La plupart de ces shells de remplacement sont disponibles sur http://www.ibiblio.org/pub/Linux/system/shells/ [http://www.ibiblio.org/pub/Linux/system/shells/]. Vérifiez que le shell que vous utilisez sait faire tourner les commandes de tous les scripts **rc** que vous incluez sur le disque d'amorce.

8.4. Nettoyez les bibliothèques et binaires

De nombreux binaires et bibliothèques restent non nettoyés (ils contiennent les informations pour le déboguage). Si vous lancez **file** sur ces fichiers, il vous indiquera *not stripped* si c'est le cas. Lorsque vous copiez des binaires sur votre système racine, une bonne habitude à prendre est d'utiliser :

```
objcopy --strip-all ORIGINE DESTINATION
```

Et lorsque vous copiez des bibliothèques :

```
objcopy --strip-debug ORIGINE DESTINATION
```

8.5. Déplacez les fichiers non essentiels vers un disque utilitaire

Si certains binaires ne sont pas immédiatement nécessaires au démarrage ou au login, vous pouvez les déplacer sur un disque utilitaire. Lisez la Section 9.2, « Construire un disque utilitaire » pour les détails. Vous pouvez aussi déplacer les modules vers un disque utilitaire.

9. Sujets divers

9.1. Système racine sans disque mémoire

La Section 4, « Construire un système racine » explique comment construire un système racine compressé chargé en mémoire lors du démarrage du système. Cette méthode qui présente beaucoup d'avantages est souvent utilisée. Néanmoins, certains systèmes possédant peu de mémoire ne peuvent se permettre d'utiliser de la RAM pour un disque mémoire, et doivent donc utiliser un système racine monté directement depuis la disquette.

De tels systèmes sont en réalité plus faciles à construire que les systèmes racines compressés car on peut les créer directement sur disquette plutôt que de passer par un autre périphérique intermédiaire, et ne nécessitent pas de compression. Nous indiquerons les différences de procédure par rapports aux instructions précédentes. Si vous choisissez cette méthode, rappelez-vous bien que vous aurez *beaucoup moins d'espace disque* disponible.

1. Calculez la taille disponible pour les fichiers racines.

   Si vous construisez un système d'amorce/racine unique, vous devez arriver à faire tenir tous les blocs du noyau ainsi que tous les blocs du système racine sur un seul disque.

2. À l'aide de **mke2fs**, créez un système racine de la bonne taille sur une disquette.

3. Remplissez le système comme décrit précédemment.

4. Après cela, démontez le système et transférez-le vers un fichier sur le disque, mais *sans le compresser*.

5. Transférez le noyau sur une disquette comme décrit précédemment. Lorsque vous calculerez le mot disque mémoire, mettez le bit 14 à 0 pour indiquer que le système racine ne doit pas être chargé en mémoire. Lancez la commande **rdev** indiquée.

6. Transférez le système racine comme précédemment.

Vous pouvez prendre quelques raccourcis. Si vous construisez un système avec deux disques, vous pouvez construire le système de fichiers racine directement sur le second disque au lieu de le transférer sur le disque dur puis à nouveau sur la disquette. De même, si vous construisez un disque d'amorce/racine unique et si vous utilisez **LILO**, vous pouvez créer un système de fichiers *unique* sur toute la disquette contenant le noyau, les fichiers de **LILO** et les fichiers racine, avant de simplement lancer **LILO** comme dernière étape.

9.2. Construire un disque utilitaire

Construire un disque utilitaire est assez facile : créez simplement un système de fichiers sur une disquette formatée et copiez les fichiers dessus. Pour l'utiliser depuis un disque d'amorce, montez-le manuellement une fois le système démarré.

Les instructions précédentes indiquent qu'un disque utilitaire peut être monté en tant que `/usr`. Dans ce cas, les binaires doivent être placés dans un répertoire `/bin` du disque utilitaire, afin d'être référencés si vous mettez `/usr/bin` dans votre chemin. Les bibliothèques supplémentaires nécessaires aux binaires sont à placer dans `/lib` sur le disque utilitaire.

Il faut penser à plusieurs choses lorsque l'on crée un disque utilitaire :

1. Ne placez pas de binaires ou de bibliothèques essentiels pour le système sur le disque utilitaire, puisqu'il ne sera montable qu'une fois le système démarré ;
2. Vous ne pouvez pas utiliser de lecteur de disquette et de lecteur de bande sur port disquette en même temps. Ce qui veut dire que si votre lecteur de bande est sur un port disquette, vous ne pourrez pas y accéder tant que votre disque utilitaire sera monté ;
3. L'accès aux fichiers du disque utilitaire sera lent.

L' Annexe D, *Exemple de contenu des répertoires d'un disque utilitaire* montre ce que peut contenir un tel disque. Voici quelques idées de fichiers qui peuvent vous être utiles : programmes de diagnostic et de manipulation de disques (**format**, **fdisk**) et systèmes de fichiers (**mke2fs**, **fsck**, **debugfs**, `isofs.o`), un éditeur de texte léger (**elvis**, **jove**), des utilitaires de compression et archivage (**gzip**, **tar**, **cpio**, **afio**), de gestion de bande (**mt**, **ftmt**, **tob**, **taper**), de communication (`ppp.o`, `slip.o`, **minicom**) et de gestion de périphériques (**setserial**, **mknod**).

10. La méthode des pros

Vous avez peut-être remarqué comme les disques d'amorce utilisés par les principales distributions comme Slackware, RedHat ou Debian paraissent plus sophistiqués que ce que décrit ce document. Les disques d'amorce de distribution professionnelles se basent sur les mêmes principes que ceux décrits ici, mais utilisent diverses astuces pour satisfaire aux besoins supplémentaires de leurs disques d'amorce. Tout d'abord, ils doivent pouvoir fonctionner sur une grande variété de matériel et doivent donc pouvoir interagir avec l'utilisateur et charger divers gestionnaires de périphériques. Ensuite, ils doivent pouvoir travailler avec beaucoup d'options d'installation différentes, de manière plus ou moins automatique. Enfin, les disques d'amorce des distributions combinent en général la possibilité d'installer le système avec celle de le réparer.

Certains disques d'amorce utilisent une fonctionnalité appelée **initrd** (*initial ramdisk*, ou *disque mémoire initial*). Cette fonctionnalité est apparue aux alentours de la version 2.0.x et permet au noyau de démarrer en deux étapes. Quand le noyau commence son démarrage, il charge une première image de disque mémoire depuis le disque d'amorce. Ce disque mémoire initial est un système racine contenant un programme à exécuter avant le chargement du vrai système racine. Ce programme inspecte en général l'environnement et/ou demande à l'utilisateur de sélectionner diverses options de démarrage, telles que le périphérique sur lequel on va trouver le vrai disque racine. En général, il charge des modules supplémentaires ne faisant pas partie du noyau. Quand ce programme initial se termine, le noyau charge la vraie image racine et continue son démarrage normalement. Pour plus d'information sur **initrd**, lisez `/usr/src/linux/Documentation/initrd.txt` et ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz [ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz].

Vous trouverez ci-dessous des résumés sur la manière dont les disques d'installation de chaque distribution semblent marcher, après étude de leurs systèmes de fichiers et/ou code source. Nous ne garantissons pas l'exactitude des informations, ni qu'elles n'ont pas changé depuis les versions indiquées.

Slackware (v.3.1) utilise un démarrage direct avec **LILO** semblable à la description de la Section 6.1, « Transfert du noyau avec LILO ». Le disque d'amorce de la Slackware affiche un message de démarrage (*Welcome to the Slackware Linux bootkernel disk!*) en utilisant le paramètre `message` de **LILO**. Ce message indique à l'utilisateur d'entrer une ligne de paramètres de démarrage si nécessaire. Après le démarrage, un système racine est chargé depuis une seconde disquette. L'utilisateur lance un script de configuration (**setup**) qui démarre l'installation. Au lieu d'utiliser un noyau modulaire, Slackware fournit un certain nombre de noyaux différents, et c'est à l'utilisateur de fournir celui qui correspond à sa configuration matérielle.

RedHat (v.4.0) utilise aussi un démarrage avec **LILO**. Il charge un disque mémoire compressé sur le premier disque, qui fait tourner une version personnalisée d'**init**. Ce programme demande quels gestionnaires utiliser puis charge des fichiers supplémentaires depuis un autre disque si nécessaire.

Debian (v.1.3) possède probablement le groupe de disques d'installation le plus sophistiqué. Il utilise le chargeur **SYSLINUX** pour choisir différentes options de chargement, puis utilise une image **initrd** pour guider l'utilisateur dans l'installation. Il semble utiliser à la fois des versions personnalisées d'**init** et du shell.

11. Créer des CD-ROM amorçables

Cette section a été écrite avec la contribution de Rizwan Mohammed Darwe (rizwan CHEZ clover-technologies POINT com).

Cette section suppose que vous êtes familier avec la procédure et le fonctionnement de l'écriture de CD sous Linux. Considérez ceci comme une référence rapide pour inclure la possibilité de démarrer à partir du CD que vous allez graver. La CD-Writing-HOWTO devrait vous donner plus de détails.

11.1. Qu'est-ce que El Torito ?

À partir des plates-formes x86, plusieurs BIOS ont commencé à accepter les CD amorçables. Les rustines pour **mkisofs** sont basés sur le standard appelé « El Torito ». En deux mots, El Torito est une spécification qui indique le format qu'un CD doit respecter, afin de pouvoir démarrer directement sur celui-ci.

La spécification « El Torito » dit que *tout* lecteur de CD-ROM devrait fonctionner (SCSI ou EIDE) si le BIOS est compatible El Torito. Sauf que cela n'a été testé qu'avec des lecteurs EIDE, car aucun contrôleur SCSI testé ne semble supporter El Torito. La carte mère doit impérativement être compatible El Torito. Comment savoir si votre carte mère est compatible El Torito ? Eh bien les cartes mères compatibles offrent le choix de démarrer à partir du disque dur, de la disquette, du réseau ou du CD-ROM.

11.2. Comment ça marche

Le standard El Torito fonctionne en faisant apparaître le lecteur de CD, à travers les appels BIOS, comme un lecteur de disquettes normal. De cette façon vous mettez simplement n'importe quelle image de la taille d'une disquette (exactement 1440 ko pour une disquette de 1,44 Mo) quelque part sur le système de fichiers ISO. Dans l'en-tête du système de fichiers ISO vous placez un pointeur vers cette image. Le BIOS va alors récupérer cette image à partir du CD et agit comme si il démarrait à partir du lecteur de disquettes. Cela permet à une disquette de démarrage **LILO**, par exemple, d'être utilisée simplement comme si c'était une vraie disquette.

En général, les 1,44 (ou 2,88 si supportés) premiers Mo du CD-ROM contiennent une image de la disquette créée par vous. Cette image est traitée comme une disquette par le BIOS et démarrée par celui-ci. (Avec comme conséquence, lors du démarrage à partir de la disquette virtuelle, que votre lecteur original *A:* (`/dev/fd0`) ne sera plus accessible, mais vous pouvez toujours essayer `/dev/fd1`).

11.3. Comment le faire marcher

D'abord, il faut créer un fichier, disons `boot.img`, qui est l'image exacte de la disquette que vous voulez démarrer via le CD-ROM. Ce doit être une disquette de démarrage 1,44 Mo. La commande suivante permet de la créer :

```
dd if=/dev/fd0 of=boot.img bs=10k count=144
```

en supposant que la disquette se trouve dans le lecteur *A:*.

Placez cette image quelque part dans la hiérarchie qui sera la source du système de fichiers iso9660. C'est une bonne idée de mettre tous les fichiers relatifs au démarrage dans leur propre répertoire (`boot/` sous la racine du système de fichiers iso9660, par exemple).

Avertissement : Votre disquette de démarrage *doit* charger un *initial ramdisk* via **LILO**, et pas le disque mémoire du noyau ! Ceci est du au fait que lorsque le noyau démarre, l'émulation BIOS du CD comme disquette est limitée et va échouer. **LILO** va charger le disque mémoire en utilisant les appels disques BIOS, et l'émulation fonctionne normalement.

La spécification El Torito requiert également la création d'un *catalogue de démarrage*. Il s'agit d'un fichier de 2048 octets qui n'a pas grand intérêt à part qu'il est nécessaire. La correction réalisée par l'auteur de **mkisofs** permet la création automatique de ce catalogue de démarrage, mais vous devez spécifier où ce catalogue doit se trouver dans la hiérarchie du système de fichiers iso9660. En général c'est une bonne idée de le mettre au même endroit que l'image de démarrage, et de l'appeler `boot.catalog`. La commande pour créer le système de fichiers iso9660 dans le fichier `bootcd.iso` est alors :

```
mkisofs -r -b boot/boot.img -c boot/boot.catalog -o bootcd.iso .
```

L'option `-b` précise l'image de démarrage à utiliser (notez que le chemin est relatif à la racine du disque iso9660), et l'option `-c` est pour le fichier catalogue de démarrage. L'option `-r` va mettre les propriétaires et droits des fichiers appropriés (voir la page de manuel de **mkisofs**). Le « `.` » à la fin dit de prendre comme source le répertoire courant.

Maintenant gravez le CD avec la commande habituelle et le voilà prêt à démarrer.

11.4. Créer des CD-ROM Win9x amorçables

La première chose à faire est de récupérer une image amorçable utilisée par le CD source. Mais vous ne pouvez pas vous contenter de monter le CD sous Linux et d'utiliser **dd** pour copier les 1440 premiers kilo-octets vers une disquette ou un fichier `boot.img`. Au lieu de cela, vous devez simplement amorcer votre système à partir du CD-ROM source.

Lorsque vous démarrez le CD Win98 vous vous retrouvez à l'invite *A:*, qui est en fait le disque mémoire. Et *D:* ou *Z:* où se trouvent tous les fichiers d'installation. En utilisant la commande DOS **diskcopy**, copiez l'image *A:* dans le vrai lecteur de disquettes, qui est maintenant *B:*. La commande suivante permet de le faire :

```
diskcopy A: B:
```

Cela fonctionne exactement comme **dd**. Vous pouvez essayer de démarrer à partir de ce disque fraîchement créé pour tester si le processus de démarrage est similaire à celui du CD source. Ensuite faites le **dd** habituel de cette disquette vers un fichier comme `boot.img` et le reste sera comme d'habitude.

12. Foire Aux Questions (FAQ)

Q :

Je démarre depuis mes disques d'amorce/racine et rien ne se passe. Que faire ?

R :

Voir la section précédente Section 7, « En cas de problème, ou l'agonie de la défaite ».

Q :

Comment fonctionne le disque d'amorce Slackware/Debian/RedHat ?

R :

Voir la section précédente Section 10, « La méthode des pros ».

Q :

Comment utiliser des disquettes de haute densité (> 1440 ko) ? Comment savoir quelles densités fonctionneront avec mon lecteur de disquette ?

R :

Voir dans la section précédente Section 8.1, « Augmentez la densité du disque » les commentaires de Alain Knaff à ce sujet. C'est la réponse la plus crédible que je connaisse.

Q :

Comment augmenter la taille de mes disques mémoire ?

R :

Cela devrait être mieux expliqué dans le texte, mais je met une réponse ici pour l'instant.

D'abord, *n'essayez pas* d'utiliser **rdev** ou **ramsize** pour faire cela, quoi qu'en disent leurs documentations. Le mot disque mémoire ne détermine plus la taille des disques mémoires.

Ensuite, gardez à l'esprit que les disques mémoires sont actuellement dynamiques ; lorsque vous définissez la taille d'un disque mémoire vous n'allouez pas de mémoire, vous précisez juste de combien il peut grandir. N'ayez pas peur de choisir une taille inutilement trop grande (par exemple 8 ou même 16 Mo). L'espace RAM n'est pas utilisé tant que vous n'en n'avez pas besoin. Vous pouvez définir ces limites de plusieurs façons différentes :

1. Utilisez le paramètre `ramdisk_size=NNN` en ligne de commande. Vous pouvez soit le rentrer à la main soit utiliser une commande comme *append="ramdisk_size=NNN"* avec **LILO** ;
2. Si vous utilisez **LILO**, vous pouvez utiliser une option du noyau comme `ramdisk=8192K` dans le fichier `lilo.conf` ;
3. Changez l'option de configuration du noyau *CONFIG_BLK_DEV_RAM_SIZE* et recompilez votre noyau.

Q :

Comment faire des CD-ROM amorçables ?

R :

Voir la section précédente Section 11, « Créer des CD-ROM amorçables ».

Q :

Comment faire des disquettes LS-120 amorçables ?

R :

Comme je n'ai pas de lecteur de disquettes LS-120, les informations qui suivent sont un résumé fourni par David Cinege du Linux Router Project.

Le LS-120 est un lecteur de disquettes IDE. Il est compatible à la fois avec les disquettes 3,5 » et les nouvelles disquettes de 120 Mo. Depuis Linux 2.0.31, celles-ci sont complètement supportées. Pour être capable de démarrer à partir de ces disquettes, vous devez avoir un BIOS qui autorise le LS-120 à être traité comme lecteur 0 (alors que les lecteurs IDE commencent normalement à 80). Si vous n'avez pas le support du BIOS, vous pouvez acheter une petite carte IDE FloppyMAX de Promise Technologies pour combler ce manque.

Le chargeur du noyau n'aime pas le LS-120, et meurt instantanément. Les disques 2m non plus ne l'aiment pas et ne démarreront pas. Les disquettes de 1,44 à 1,74 Mo fonctionnent bien. **SYSLINUX** fonctionne avec les disquettes de 120 Mo à partir de la version 1.32. Vous auriez intérêt à partitionner la disquette et utiliser ext2 ou minix, au lieu de **SYSLINUX**, sauf si vous avez besoin d'une compatibilité MS-DOS.

LILO fonctionne bien avec des disquettes de 120 Mo. Voici un `lilo.conf` simple :

```
boot=/dev/hda
compact
```

```
          disk=/dev/hda bios=0
          install=/floppy/boot.b
          map=/floppy/map
          image=/floppy/linux
          label=Linux
          append="load_ramdisk=1"
          initrd=/floppy/root.bin
          ramdisk=8192
```

La ligne *disk=/dev/hda bios=0* est la ruse pour démarrer à partir du LS-120.

Q :

Comment faire un disque d'amorce avec un gestionnaire pour XYZ ?

R :

Le plus simple est d'obtenir un noyau Slackware depuis le site miroir de Slackware le plus proche. Les noyaux Slackware sont des noyaux génériques contenant le plus de gestionnaires pour le plus de périphériques différents possibles. Si vous avez un contrôleur SCSI ou IDE, vous avez de bonnes chances de trouver un gestionnaire correspondant dans le noyau Slackware.

Allez dans le répertoire `a1` et sélectionnez un noyau SCSI ou IDE suivant votre type de contrôleur. Vérifiez dans le fichier `xxxxkern.cfg` correspondant au noyau choisi qu'il contient bien les gestionnaires que vous voulez. Si c'est le cas, le noyau correspondant devrait pouvoir démarrer votre ordinateur. Récupérez le fichier `xxxxkern.tgz` et copiez-le sur votre disquette d'amorce comme indiqué dans la section sur la fabrication des disques d'amorce.

Vous devez ensuite vérifier le périphérique racine indiqué dans le noyau, en utilisant la commande :

```
rdev zImage
```

rdev vous montrera alors le périphérique actuellement configuré dans le noyau. Si ce n'est pas celui que vous voulez, utilisez **rdev** pour le changer. Par exemple, le noyau que j'ai essayé pointait sur `/dev/sda2`, mais ma partition racine SCSI est sur `/dev/sda8`. Pour utiliser une disquette racine, vous devrez lancer la commande :

```
rdev zImage /dev/fd0
```

Si vous voulez aussi savoir comment configurer un disque racine Slackware, cela dépasse le cadre de ce HOWTO, et je vous suggère donc de consulter le Guide d'installation de Linux ou de récupérer la distribution Slackware. Voir l'Annexe A, *Ressources et pointeurs* de ce HOWTO.

Q :

Comment mettre à jour le noyau de ma disquette d'amorce ?

R :

Copiez simplement le noyau sur votre disquette d'amorce à l'aide de la commande **dd** s'il s'agit d'une disquette d'amorce sans système de fichier, ou par la commande **cp** pour un disque d'amorce/racine. Reportez-vous à la Section 3.1, « Démarrage » de ce HOWTO pour les détails de création d'un disque d'amorce. Le processus décrit s'applique aussi bien à la mise à jour d'un noyau sur le disque d'amorce.

Q :

Comment mettre à jour ma disquette racine avec de nouveaux fichiers ?

R :

Le plus simple est de recopier le système de fichiers depuis le disque racine vers le PÉRIPHÉRIQUE que vous avez utilisé (comme dans la section précédente Section 4.2, « Création du système de fichiers »). Montez ensuite le système de fichiers et modifiez-le. Vous devez vous souvenir d'où partait votre système racine et du nombre de blocs qu'il occupait :

```
dd if=/dev/fd0 bs=1k skip=DEBUTRACINE count=BLOCS | \
        gunzip > PÉRIPHÉRIQUE
mount -t ext2 PÉRIPHÉRIQUE /mnt
```

Une fois les modifications effectuées, recommencez comme précédemment (dans la Section 4.7, « C'est dans la poche ») et retransférez le système racine sur le disque. Vous ne devriez pas avoir à retransférer le noyau ou à recalculer le mot disque mémoire si vous ne changez pas la position de départ du nouveau système de fichiers.

Q :

Comment retirer **LILO** pour pouvoir redémarrer DOS ?

R :

Ce n'est pas réellement un problème de disque d'amorce, mais il est souvent posé. Sous Linux, vous pouvez lancer :

```
/sbin/lilo -u
```

Vous pouvez aussi utiliser la commande **dd** pour copier la sauvegarde effectuée par **LILO** sur le secteur d'amorce. Reportez-vous à la documentation de **LILO** si vous voulez essayer.

Sous DOS et Windows vous pouvez utiliser la commande DOS :

```
FDISK /MBR
```

MBR signifie Master Boot Record (Enregistrement d'amorce Maître), et il remplace le secteur de démarrage avec une version propre du DOS, sans modifier la table de partitions. Certains puristes n'apprécient pas cette méthode, mais même l'auteur de **LILO**, Werner Almesberger, le suggère. C'est facile et ça marche.

Q :

Comment puis-je démarrer si j'ai perdu mon noyau *et* mon disque d'amorce ?

R :

Si vous n'avez pas de disque d'amorce sous la main, le plus simple est d'obtenir un noyau Slackware pour votre type de contrôleur de disque (IDE ou SCSI) comme décrit précédemment dans « Comment faire un disque d'amorce avec un gestionnaire pour XYZ ? ». Vous pouvez alors démarrer votre ordinateur avec ce noyau, puis réparer les dommages éventuels.

Le noyau que vous récupérerez peut ne pas avoir comme périphérique racine ce que vous souhaitez comme disque et partition. Par exemple, le noyau générique SCSI de Slackware utilise `/dev/sda2` comme périphérique racine, alors que ma partition racine Linux se trouve être `/dev/sda8`. Dans ce cas il faut changer le périphérique racine.

Vous pouvez changer les paramètres de périphérique racine et disque mémoire du noyau même si vous n'avez que le noyau, et un autre système d'exploitation tel que DOS.

rdev modifie les paramètres du noyau en changeant les valeurs à un décalage fixé dans le fichier du noyau, et vous pouvez donc faire de même si vous avez un éditeur hexadécimal disponible sous quelque système d'exploitation fonctionnant encore — par exemple, Norton Utilities Disk Editor sous DOS. Vous devez alors vérifier puis éventuellement modifier les valeurs dans le noyau, aux décalages suivants :

```
HEX     DEC  DESCRIPTION
0x01F8  504  Octet de poids faible du mot disque mémoire
0x01F9  505  Octet de poids fort du mot disque mémoire
0x01FC  508  Numéro mineur du périphérique racine : voir ci-dessous
0X01FD  509  Numéro majeur du périphérique racine : voir ci-dessous
```

L'interprétation du mot disque mémoire était décrite dans la précédente Section 6.3, « Mise en place du mot disque mémoire ».

Les numéros majeurs et mineurs de périphérique doivent correspondre au périphérique à partir duquel le système racine sera monté. Certaines valeurs utiles parmi lesquelles vous pouvez choisir sont :

```
DEVICE        MAJEUR MINEUR
/dev/fd0           2      0   1er lecteur de disquette
/dev/hda1          3      1   partition 1 sur le 1er disque IDE
/dev/sda1          8      1   partition 1 sur le 1er disque SCSI
/dev/sda8          8      8   partition 8 sur le 1er disque SCSI
```

Une fois ces valeurs mises en place, vous pouvez écrire le fichier sur une disquette en utilisant soit Norton Utilities Disk Editor, soit un programme appelé **rawrite.exe**. Ce programme est inclus dans toutes les distributions. C'est un programme DOS qui écrit directement un fichier sur le disque, en commençant à partir du secteur d'amorce, au lieu de l'écrire dans le système de fichiers. Si vous utilisez Norton Utilities, vous devez écrire le fichier sur un disque physique en commençant au début du disque.

Q :

Comment faire des copies supplémentaires des disquettes d'amorce/racine ?

R :

Les supports magnétiques se détériorant avec le temps, vous devriez conserver plusieurs copies de votre disque de secours, au cas où l'original ne serait plus lisible.

Le plus simple pour copier une disquette quelle qu'elle soit, y compris une disquette d'amorce ou utilitaire, est d'utiliser la commande **dd** pour copier le contenu de la disquette originale vers un fichier de votre disque dur, puis de réutiliser la même commande pour recopier le fichier vers une nouvelle disquette. Notez que vous n'avez pas besoin de monter la disquette, et ne devriez pas le faire, car **dd** utilise l'interface directe du périphérique.

Pour copier l'original, entrez la commande :

```
dd if=NOMPÉRIPHÉRIQUE of=NOMFICHIER
```

où *`NOMPÉRIPHÉRIQUE`* est le nom du périphérique du lecteur de disquette et *`NOMFICHIER`* le nom du fichier de sortie (sur le disque dur).

Ne pas mettre le paramètre `count` permet à **dd** de copier la disquette en entier (2880 blocs en haute densité).

Pour recopier le fichier résultant sur une nouvelle disquette, insérez celle-ci et entrez la commande inverse :

```
dd if=NOMFICHIER of=NOMPÉRIPHÉRIQUE
```

À noter que la discussion précédente suppose que vous n'avez qu'un seul lecteur de disquette. Si vous en avez deux du même type, vous pouvez copier les disquettes à l'aide d'une commande du type :

```
dd if=/dev/fd0 of=/dev/fd1
```

Q :

Comment puis-je démarrer sans avoir à taper « ahaxxxx=nn,nn,nn » à chaque fois ?

R :

Quand un périphérique disque ne peut pas être détecté automatiquement, il faut fournir au noyau une chaîne de paramètres de commande du périphérique, telle que :

```
aha152x=0x340,11,3,1
```

Cette chaîne peut être fournie de différentes manières grâce à **LILO** :

- En l'entrant sur la ligne de commande à chaque démarrage du système avec **LILO**. C'est assez ennuyeux ;
- En utilisant le mot clé *lock* de **LILO** pour lui faire stocker la ligne de commande comme ligne de commande par défaut, ce qui fera utiliser à **LILO** les mêmes options à chaque démarrage ;
- En utilisant la directive *append*= dans le fichier de configuration de **LILO**. Attention à encadrer la chaîne de paramètres avec des guillemets.

Par exemple, une ligne de commande utilisant la chaîne ci-dessus serait :

```
zImage aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

Cela passerait la chaîne de paramètres pour le périphérique tout en demandant au noyau d'utiliser `/dev/sda1` comme périphérique racine et de sauvegarder la ligne de commande pour la réutiliser pour tous les démarrages futurs.

Un exemple de directive *APPEND* peut être :

```
APPEND = "aha152x=0x340,11,3,1"
```

Attention, la chaîne de paramètres ne doit PAS être entourée de guillemets sur la ligne de commande, mais DOIT l'être dans la directive *APPEND*.

Notez aussi que pour que la chaîne de paramètres soit utilisée, le noyau doit contenir le gestionnaire pour ce type de disque. Si ce n'est pas le cas, personne n'écoutera la chaîne de paramètres, et vous devrez reconstruire le noyau pour inclure le gestionnaire requis. Pour plus de détails sur la reconstruction du noyau, rendez-vous dans `/usr/src/linux` et lisez le `README`, ou lisez la FAQ Linux et le HOWTO Installation. Vous pouvez aussi obtenir un noyau générique pour votre type de disque et l'installer.

Il est fortement recommandé aux lecteurs de lire la documentation de **LILO** avant de faire des expériences d'installation de **LILO**. Une utilisation imprudente de la directive *BOOT* peut endommager des partitions.

Q :

Au démarrage, j'obtiens l'erreur « A: cannot execute B ». Pourquoi ?

R :

Il existe plusieurs utilitaires qui référencent en dur le nom d'autres programmes. Ça n'arrive pas tout le temps, mais cela peut expliquer pourquoi un exécutable peut ne pas être trouvé sur votre système même si vous l'y voyez. Vous pouvez vérifier si un programme donné est référencé en dur dans un autre en utilisant la commande **strings** et en passant son résultat par **grep**.

On trouve comme exemples connus de référence en dur :

- **shutdown** dans certaines versions référence `/etc/reboot` en dur, et **reboot** doit donc être placé dans le répertoire `/etc` ;
- **init** a posé des problèmes à au moins une personne, pour qui le noyau ne trouvait pas **init**.

Pour corriger ces problèmes, vous pouvez soit déplacer les programmes vers le répertoire attendu, soit changer les fichiers de configuration (par exemple `inittab`) pour référencer le bon répertoire. En cas de doute, mettez les programmes dans le même répertoire que sur votre disque dur, et utilisez les mêmes fichiers `inittab` et `/etc/rc.d` que sur celui-ci.

Q :

Mon noyau gère les disques mémoires, mais les initialise à 0 ko

R :

Quand cela arrive, un message du noyau apparaîtra au moment du démarrage, du type :

```
Ramdisk driver initialized : 16 ramdisks of 0K size
```

C'est probablement parce que la taille a été fixée par les paramètres du noyau à 0 au moment du démarrage. Cela peut être dû à un paramètre oublié dans le fichier de configuration de **LILO** :

```
ramdisk= 0
```

Certaines vieilles distributions l'incluaient dans des exemples de fichiers de configuration de **LILO**, et servaient à écraser les paramètres antérieurs du noyau. Si vous trouvez une telle ligne, supprimez-la.

Attention, si vous essayez d'utiliser un disque mémoire dont la taille est de 0 ko, le comportement est imprévisible et peut conduire à une panique (panic) du noyau.

Ressources et pointeurs

Lorsque vous récupérez un paquetage, prenez toujours la dernière version, sauf si vous avez de bonnes raisons pour ne pas le faire.

1. Disques d'amorce préfabriqués

Ce sont les sources des disques d'amorce des distributions. *Merci d'utiliser un site miroir pour réduire la charge sur ces machines.*

- Disques d'amorce Slackware [http://distro.ibiblio.org/pub/Linux/distributions/slackware/slackware-current/bootdisks/], disques racines [http://distro.ibiblio.org/pub/Linux/distributions/slackware/slackware-current/rootdisks/] et sites miroirs Slackware [http://www.slackware.com/getslack/] ;

- Disques d'amorce RedHat [ftp://ftp.redhat.com/pub/redhat/linux/current/en/os/i386/images/] et sites miroirs Red Hat [http://www.redhat.com/mirrors.html] ;
- Disques d'amorce Debian [ftp://ftp.debian.org/debian/dists/stable/main/disks-i386/current/] et sites miroirs Debian [ftp://ftp.debian.org/debian/README.mirrors.html].
- Téléchargements Mandrake [http://www.linux-mandrake.com/en/ftp.php3].

En plus des disques d'amorce des distributions, les images de disques de secours suivantes sont disponibles. Sauf précision contraire, elles sont disponibles dans le répertoire http://www.ibiblio.org/pub/Linux/system/recovery/!INDEX.html [http://www.ibiblio.org/pub/Linux/system/recovery/!INDEX.html].

- RIP est un système de démarrage/secours qui existe en plusieurs versions : une qui va sur une disquette de 1,44 Mo et une qui va sur un CD-ROM. Il supporte les grands fichiers et plusieurs programmes pour la maintenance des disques et le secours. Il supporte ext2, ext3, iso9660, msdos, ntfs, reiserfs, ufs et vfat. RIP est disponible sur http://www.tux.org/pub/people/kent-robotti/looplinux/rip/index.html ;
- tomsrtbt, par Tom Oehser, est un disque d'amorce/racine unique à base de noyau 2.0, avec de nombreux programmes de support et fonctionnalités. Il support IDE, SCSI, les bandes, les adaptateurs réseaux, PCMCIA et plus encore. Environ 100 programmes utilitaires et autres outils sont inclus, pour réparer et récupérer les disques. Le paquetage contient aussi des scripts pour désassembler et reconstruire les images afin de pouvoir ajouter des compléments si nécessaire ;
- rescue02, par John Comyns, est un disque de secours à base de noyau 1.3.84, qui supporte IDE, Adaptec 1542 et NCR53C7,8xx. Il est à base de binaires ELF mais contient suffisamment de commandes pour être utilisé sur n'importe quel système. Certains modules peuvent être chargés après le démarrage pour d'autres cartes SCSI. Il ne fonctionnera probablement pas sur les systèmes avec 4 Mo de RAM car il utilise un disque mémoire de 3 Mo ;
- resque_disk-2.0.22, par Sergei Viznyuk, est un disque d'amorce/racine complet basé sur le noyau 2.0.22, comprenant le support pour IDE, de nombreux contrôleurs SCSI, et ELF/a.out. Il contient aussi nombre de modules et d'utilitaires pour réparer et récupérer un disque dur ;
- les images de cramdisk, à base de noyau 2.0.33, disponibles pour machines à 4 et 8 Mo de mémoire. Elles contiennent l'émulation mathématique et le réseau (PPP et script dialin, NE2000, 3C509), ou le support pour lecteur ZIP sur port parallèle. Ces images de disquettes peuvent démarrer un 386 avec 4 Mo de RAM. Le support de MSDOS est inclus, ce qui fait que vous pouvez les récupérer sur le réseau vers une partition DOS.

  http://www.ibiblio.org/pub/Linux/system/recovery/images

2. Paquetages de secours

Plusieurs paquetages de création de disques de secours existent sur www.ibiblio.org. Vous précisez à ces paquetages un ensemble de fichiers à inclure, et le logiciel automatise (à divers degrés) la création d'un disque d'amorce. Voir http://www.ibiblio.org/pub/Linux/system/recovery/!INDEX.html [http://www.ibiblio.org/pub/Linux/system/recovery/!INDEX.html] pour plus d'informations. *Vérifiez bien les dates des fichiers* : certains paquetages n'ont pas été mis à jour depuis des années et ne supportent pas la création d'un système racine compressé sur disque mémoire. À notre connaissance, Yard est le seul paquetage le permettant.

3. LILO : le chargeur Linux

Écrit par Werner Almesberger. Excellent chargeur d'amorce, dont la documentation comprend des informations sur le contenu du secteur d'amorce et les premières étapes du processus de démarrage.

Ftp depuis ftp://tsx-11.mit.edu/pub/linux/packages/lilo/. Il est aussi disponible sur Metalab et ses miroirs.

4. Utilisation du disque mémoire

La documentation fournie avec le noyau Linux contient une excellente description de la manière dont fonctionne le nouveau code de disque mémoire. Voir `/usr/src/linux/Documentation/ramdisk.txt`. C'est écrit par Paul Gortmaker, et cela comprend une section sur la création d'un disque mémoire compressé.

5. Le processus de démarrage de Linux

Pour plus de détails sur le processus de démarrage de Linux, voici quelques pointeurs :

- Le Linux System Administrators' Guide (Guide des Administrateurs Systèmes Linux) contient une section sur le démarrage. Voir http://www.traduc.org/docs/guides/lecture/sag/ ;
- La « Technical overview » (Description technique succincte) de **LILO** http://www.ibiblio.org/pub/Linux/system/boot/lilo/lilo-t-21.ps.gz décrit d'une manière extrêmement poussée le processus de démarrage, d'un point de vue technique et bas niveau, jusqu'au moment où le noyau est lancé ;
- Le code source est le guide ultime. Ci-dessous se trouvent quelques fichiers du noyau relatifs au processus de démarrage. Si vous avez le code source du noyau Linux, vous pouvez les trouver sous `/usr/src/linux` sur votre machine ; sinon, Shigio Yamaguchi `<shigio CHEZ wafu POINT netgate POINT net>` a un très sympathique navigateur hypertexte pour le noyau à http://www.tamacom.com/tour/linux/index.html. Voici quelques fichiers correspondants :

arch/i386/boot/bootsect.S,setup.S	Contient le code assembleur pour le secteur d'amorce.
arch/i386/boot/compressed/misc.c	Contient le code pour décompresser le noyau.
arch/i386/kernel/	Répertoire contenant le code d'initialisation du noyau. setup.c contient le mot disque mémoire.
drivers/block/rd.c	Contient le gestionnaire de disque mémoire. Les procédures rd_load et rd_load_image chargent des blocs depuis un périphérique vers un disque mémoire. La procédure identify_ramdisk_image détermine le type de système de fichiers trouvé, et s'il est compressé.

Codes d'erreur du démarrage de LILO

Les questions sur ces erreurs sont posées si souvent sur Usenet que nous les incluons ici en tant que service public. Ce résumé est extrait de la Documentation Utilisateur de **LILO** de Werner Almesberger, disponible sur http://www.ibiblio.org/pub/Linux/system/boot/lilo/lilo-u-21.ps.gz.

Quand **LILO** se charge, il affiche le mot **LILO**. Chaque lettre est imprimée avant ou après l'exécution d'une action spécifique. Si **LILO** échoue à un moment donné, les lettres affichées jusque là peuvent être utilisées pour identifier le problème.

(rien)	Aucun morceau de **LILO** n'a été chargé. Soit **LILO** n'est pas installé, soit la partition sur laquelle son secteur d'amorce se trouve n'est pas active.

L Le premier morceau du chargeur d'amorce a été chargé et démarré, mais il ne peut charger le second morceau. Les codes d'erreur à deux chiffres indiquent le type de problème. (Voir également la section « Codes d'erreur disque ».) Ce cas indique en général une panne de périphérique ou une incohérence de géométrie (c'est à dire de mauvais paramètres disques).

LI Le premier morceau du chargeur d'amorce a pu charger le second morceau, mais n'a pas réussi à l'exécuter. Cela peut être causé par une incohérence de géométrie ou par le déplacement de /boot/boot.b sans lancer l'installateur de carte.

LIL Le second morceau du chargeur d'amorce a été démarré, mais il ne trouve pas la table de descripteurs dans le fichier carte. C'est en général dû à une panne de périphérique ou une incohérence de géométrie.

LIL? Le second morceau du chargeur d'amorce a été chargé à un adresse incorrecte. C'est en général causé par une subtile incohérence de géométrie, ou par le déplacement de `/boot/boot.b` sans lancer l'installateur de carte.

LIL- La table de descripteurs est corrompue. Cela peut être dû à une incohérence de géométrie ou au déplacement de `/boot/map` sans lancer l'installateur.

LILO Tous les éléments de **LILO** ont été correctement chargés.

Si le BIOS signale une erreur lorsque **LILO** essaye de charger une image d'amorce, le code d'erreur correspondant est affiché. Ces codes vont de *0x00* à *0xbb*. Reportez-vous au Guide Utilisateur de **LILO** pour leur explication.

Exemple de contenu de répertoires sur un disque racine

Voici le contenu d'un exemple de système racine et d'une disquette utilitaire.

```
/:
drwx--x--x   2 root     root          1024 Nov  1 15:39 bin
drwx--x--x   2 root     root          4096 Nov  1 15:39 dev
drwx--x--x   3 root     root          1024 Nov  1 15:39 etc
drwx--x--x   4 root     root          1024 Nov  1 15:39 lib
drwx--x--x   5 root     root          1024 Nov  1 15:39 mnt
drwx--x--x   2 root     root          1024 Nov  1 15:39 proc
drwx--x--x   2 root     root          1024 Nov  1 15:39 root
drwx--x--x   2 root     root          1024 Nov  1 15:39 sbin
drwx--x--x   2 root     root          1024 Nov  1 15:39 tmp
drwx--x--x   7 root     root          1024 Nov  1 15:39 usr
drwx--x--x   5 root     root          1024 Nov  1 15:39 var
/bin:
-rwx--x--x   1 root     root         62660 Nov  1 15:39 ash
-rwx--x--x   1 root     root          9032 Nov  1 15:39 cat
-rwx--x--x   1 root     root         10276 Nov  1 15:39 chmod
-rwx--x--x   1 root     root          9592 Nov  1 15:39 chown
-rwx--x--x   1 root     root         23124 Nov  1 15:39 cp
-rwx--x--x   1 root     root         23028 Nov  1 15:39 date
-rwx--x--x   1 root     root         14052 Nov  1 15:39 dd
-rwx--x--x   1 root     root         14144 Nov  1 15:39 df
-rwx--x--x   1 root     root         69444 Nov  1 15:39 egrep
-rwx--x--x   1 root     root           395 Nov  1 15:39 false
-rwx--x--x   1 root     root         69444 Nov  1 15:39 fgrep
-rwx--x--x   1 root     root         69444 Nov  1 15:39 grep
-rwx--x--x   3 root     root         45436 Nov  1 15:39 gunzip
-rwx--x--x   3 root     root         45436 Nov  1 15:39 gzip
-rwx--x--x   1 root     root          8008 Nov  1 15:39 hostname
-rwx--x--x   1 root     root         12736 Nov  1 15:39 ln
```

```
-rws--x--x   1 root     root        15284 Nov  1 15:39 login
-rwx--x--x   1 root     root        29308 Nov  1 15:39 ls
-rwx--x--x   1 root     root         8268 Nov  1 15:39 mkdir
-rwx--x--x   1 root     root         8920 Nov  1 15:39 mknod
-rwx--x--x   1 root     root        24836 Nov  1 15:39 more
-rws--x--x   1 root     root        37640 Nov  1 15:39 mount
-rwx--x--x   1 root     root        12240 Nov  1 15:39 mt
-rwx--x--x   1 root     root        12932 Nov  1 15:39 mv
-r-x--x--x   1 root     root        12324 Nov  1 15:39 ps
-rwx--x--x   1 root     root         5388 Nov  1 15:39 pwd
-rwx--x--x   1 root     root        10092 Nov  1 15:39 rm
lrwxrwxrwx   1 root     root            3 Nov  1 15:39 sh -> ash
-rwx--x--x   1 root     root        25296 Nov  1 15:39 stty
-rws--x--x   1 root     root        12648 Nov  1 15:39 su
-rwx--x--x   1 root     root         4444 Nov  1 15:39 sync
-rwx--x--x   1 root     root        19712 Nov  1 15:39 touch
-rwx--x--x   1 root     root          395 Nov  1 15:39 true
-rws--x--x   1 root     root        19084 Nov  1 15:39 umount
-rwx--x--x   1 root     root         5368 Nov  1 15:39 uname
-rwx--x--x   3 root     root        45436 Nov  1 15:39 zcat
/dev:
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 cdrom -> cdu31a
brw-rw-r--   1 root     root      15,   0 May  5  1998 cdu31a
crw-------   1 root     root       4,   0 Nov  1 15:29 console
crw-rw-rw-   1 root     uucp       5,  64 Sep  9 19:46 cua0
crw-rw-rw-   1 root     uucp       5,  65 May  5  1998 cua1
crw-rw-rw-   1 root     uucp       5,  66 May  5  1998 cua2
crw-rw-rw-   1 root     uucp       5,  67 May  5  1998 cua3
brw-rw----   1 root     floppy     2,   0 Aug  8 13:54 fd0
brw-rw----   1 root     floppy     2,  36 Aug  8 13:54 fd0CompaQ
brw-rw----   1 root     floppy     2,  84 Aug  8 13:55 fd0D1040
brw-rw----   1 root     floppy     2,  88 Aug  8 13:55 fd0D1120
brw-rw----   1 root     floppy     2,  12 Aug  8 13:54 fd0D360
brw-rw----   1 root     floppy     2,  16 Aug  8 13:54 fd0D720
brw-rw----   1 root     floppy     2, 120 Aug  8 13:55 fd0D800
brw-rw----   1 root     floppy     2,  32 Aug  8 13:54 fd0E2880
brw-rw----   1 root     floppy     2, 104 Aug  8 13:55 fd0E3200
brw-rw----   1 root     floppy     2, 108 Aug  8 13:55 fd0E3520
brw-rw----   1 root     floppy     2, 112 Aug  8 13:55 fd0E3840
brw-rw----   1 root     floppy     2,  28 Aug  8 13:54 fd0H1440
brw-rw----   1 root     floppy     2, 124 Aug  8 13:55 fd0H1600
brw-rw----   1 root     floppy     2,  44 Aug  8 13:55 fd0H1680
brw-rw----   1 root     floppy     2,  60 Aug  8 13:55 fd0H1722
brw-rw----   1 root     floppy     2,  76 Aug  8 13:55 fd0H1743
brw-rw----   1 root     floppy     2,  96 Aug  8 13:55 fd0H1760
brw-rw----   1 root     floppy     2, 116 Aug  8 13:55 fd0H1840
brw-rw----   1 root     floppy     2, 100 Aug  8 13:55 fd0H1920
lrwxrwxrwx   1 root     root            7 Nov  1 15:39 fd0H360 -> fd0D360
lrwxrwxrwx   1 root     root            7 Nov  1 15:39 fd0H720 -> fd0D720
brw-rw----   1 root     floppy     2,  52 Aug  8 13:55 fd0H820
brw-rw----   1 root     floppy     2,  68 Aug  8 13:55 fd0H830
brw-rw----   1 root     floppy     2,   4 Aug  8 13:54 fd0d360
brw-rw----   1 root     floppy     2,   8 Aug  8 13:54 fd0h1200
brw-rw----   1 root     floppy     2,  40 Aug  8 13:54 fd0h1440
brw-rw----   1 root     floppy     2,  56 Aug  8 13:55 fd0h1476
brw-rw----   1 root     floppy     2,  72 Aug  8 13:55 fd0h1494
brw-rw----   1 root     floppy     2,  92 Aug  8 13:55 fd0h1600
brw-rw----   1 root     floppy     2,  20 Aug  8 13:54 fd0h360
brw-rw----   1 root     floppy     2,  48 Aug  8 13:55 fd0h410
brw-rw----   1 root     floppy     2,  64 Aug  8 13:55 fd0h420
brw-rw----   1 root     floppy     2,  24 Aug  8 13:54 fd0h720
brw-rw----   1 root     floppy     2,  80 Aug  8 13:55 fd0h880
brw-rw----   1 root     disk       3,   0 May  5  1998 hda
brw-rw----   1 root     disk       3,   1 May  5  1998 hda1
brw-rw----   1 root     disk       3,   2 May  5  1998 hda2
brw-rw----   1 root     disk       3,   3 May  5  1998 hda3
brw-rw----   1 root     disk       3,   4 May  5  1998 hda4
brw-rw----   1 root     disk       3,   5 May  5  1998 hda5
brw-rw----   1 root     disk       3,   6 May  5  1998 hda6
```

```
brw-rw----   1 root     disk       3,  64 May  5  1998 hdb
brw-rw----   1 root     disk       3,  65 May  5  1998 hdb1
brw-rw----   1 root     disk       3,  66 May  5  1998 hdb2
brw-rw----   1 root     disk       3,  67 May  5  1998 hdb3
brw-rw----   1 root     disk       3,  68 May  5  1998 hdb4
brw-rw----   1 root     disk       3,  69 May  5  1998 hdb5
brw-rw----   1 root     disk       3,  70 May  5  1998 hdb6
crw-r-----   1 root     kmem       1,   2 May  5  1998 kmem
crw-r-----   1 root     kmem       1,   1 May  5  1998 mem
lrwxrwxrwx   1 root     root           12 Nov  1 15:39 modem -> ttyS1
lrwxrwxrwx   1 root     root           12 Nov  1 15:39 mouse -> psaux
crw-rw-rw-   1 root     root       1,   3 May  5  1998 null
crwxrwxrwx   1 root     root      10,   1 Oct  5 20:22 psaux
brw-r-----   1 root     disk       1,   1 May  5  1998 ram
brw-rw----   1 root     disk       1,   0 May  5  1998 ram0
brw-rw----   1 root     disk       1,   1 May  5  1998 ram1
brw-rw----   1 root     disk       1,   2 May  5  1998 ram2
brw-rw----   1 root     disk       1,   3 May  5  1998 ram3
brw-rw----   1 root     disk       1,   4 May  5  1998 ram4
brw-rw----   1 root     disk       1,   5 May  5  1998 ram5
brw-rw----   1 root     disk       1,   6 May  5  1998 ram6
brw-rw----   1 root     disk       1,   7 May  5  1998 ram7
brw-rw----   1 root     disk       1,   8 May  5  1998 ram8
brw-rw----   1 root     disk       1,   9 May  5  1998 ram9
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 ramdisk -> ram0
***  Je n'ai inclus de périphériques que pour les partitions IDE que
***  j'utilise. Si vous utilisez du SCSI, prenez les périphériques
***  /dev/sdXX à la place.
crw-------   1 root     root       4,   0 May  5  1998 tty0
crw--w----   1 root     tty        4,   1 Nov  1 15:39 tty1
crw-------   1 root     root       4,   2 Nov  1 15:29 tty2
crw-------   1 root     root       4,   3 Nov  1 15:29 tty3
crw-------   1 root     root       4,   4 Nov  1 15:29 tty4
crw-------   1 root     root       4,   5 Nov  1 15:29 tty5
crw-------   1 root     root       4,   6 Nov  1 15:29 tty6
crw-------   1 root     root       4,   7 May  5  1998 tty7
crw-------   1 root     tty        4,   8 May  5  1998 tty8
crw-------   1 root     tty        4,   9 May  8 12:57 tty9
crw-rw-rw-   1 root     root       4,  65 Nov  1 12:17 ttyS1
crw-rw-rw-   1 root     root       1,   5 May  5  1998 zero
/etc:
-rw-------   1 root     root          164 Nov  1 15:39 conf.modules
-rw-------   1 root     root          668 Nov  1 15:39 fstab
-rw-------   1 root     root           71 Nov  1 15:39 gettydefs
-rw-------   1 root     root          389 Nov  1 15:39 group
-rw-------   1 root     root          413 Nov  1 15:39 inittab
-rw-------   1 root     root           65 Nov  1 15:39 issue
-rw-r--r--   1 root     root          746 Nov  1 15:39 ld.so.cache
-rw-------   1 root     root           32 Nov  1 15:39 motd
-rw-------   1 root     root          949 Nov  1 15:39 nsswitch.conf
drwx--x--x   2 root     root         1024 Nov  1 15:39 pam.d
-rw-------   1 root     root          139 Nov  1 15:39 passwd
-rw-------   1 root     root          516 Nov  1 15:39 profile
-rwx--x--x   1 root     root          387 Nov  1 15:39 rc
-rw-------   1 root     root           55 Nov  1 15:39 shells
-rw-------   1 root     root          774 Nov  1 15:39 termcap
-rw-------   1 root     root           78 Nov  1 15:39 ttytype
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 utmp -> ../var/run/utmp
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 wtmp -> ../var/log/wtmp
/etc/pam.d:
-rw-------   1 root     root          356 Nov  1 15:39 other
/lib:
-rwxr-xr-x   1 root     root        45415 Nov  1 15:39 ld-2.0.7.so
lrwxrwxrwx   1 root     root           11 Nov  1 15:39 ld-linux.so.2 -> ld-2.0.7.so
-rwxr-xr-x   1 root     root       731548 Nov  1 15:39 libc-2.0.7.so
lrwxrwxrwx   1 root     root           13 Nov  1 15:39 libc.so.6 -> libc-2.0.7.so
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcom_err.so.2 -> libcom_err.so.2.0
-rwxr-xr-x   1 root     root         6209 Nov  1 15:39 libcom_err.so.2.0
-rwxr-xr-x   1 root     root       153881 Nov  1 15:39 libcrypt-2.0.7.so
```

```
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcrypt.so.1 -> libcrypt-2.0.7.so
-rwxr-xr-x   1 root     root        12962 Nov  1 15:39 libdl-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libdl.so.2 -> libdl-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libext2fs.so.2 -> libext2fs.so.2.4
-rwxr-xr-x   1 root     root        81382 Nov  1 15:39 libext2fs.so.2.4
-rwxr-xr-x   1 root     root        25222 Nov  1 15:39 libnsl-2.0.7.so
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libnsl.so.1 -> libnsl-2.0.7.so
-rwx--x--x   1 root     root       178336 Nov  1 15:39 libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           21 Nov  1 15:39 libnss_files.so.1 -> libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libpam.so.0 -> libpam.so.0.64
-rwxr-xr-x   1 root     root        26906 Nov  1 15:39 libpam.so.0.64
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libpam_misc.so.0 -> libpam_misc.so.0.64
-rwxr-xr-x   1 root     root         7086 Nov  1 15:39 libpam_misc.so.0.64
-r-xr-xr-x   1 root     root        35615 Nov  1 15:39 libproc.so.1.2.6
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libpwdb.so.0 -> libpwdb.so.0.54
-rw-r-r---   1 root     root       121899 Nov  1 15:39 libpwdb.so.0.54
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libtermcap.so.2 -> libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12041 Nov  1 15:39 libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12874 Nov  1 15:39 libutil-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libutil.so.1 -> libutil-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libuuid.so.1 -> libuuid.so.1.1
-rwxr-xr-x   1 root     root         8039 Nov  1 15:39 libuuid.so.1.1
drwx--x--x   3 root     root         1024 Nov  1 15:39 modules
drwx--x--x   2 root     root         1024 Nov  1 15:39 security
/lib/modules:
drwx--x--x   4 root     root         1024 Nov  1 15:39 2.0.35
/lib/modules/2.0.35:
drwx--x--x   2 root     root         1024 Nov  1 15:39 block
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom
/lib/modules/2.0.35/block:
drwx------   1 root     root         7156 Nov  1 15:39 loop.o
/lib/modules/2.0.35/cdrom:
drwx------   1 root     root        24108 Nov  1 15:39 cdu31a.o
/lib/security:
-rwx--x--x   1 root     root         8771 Nov  1 15:39 pam_permit.so
***  Répertoires bases pour les montages
/mnt:
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom
drwx--x--x   2 root     root         1024 Nov  1 15:39 floppy
/proc:
/root:
-rw-------   1 root     root          176 Nov  1 15:39 .bashrc
-rw-------   1 root     root          182 Nov  1 15:39 .cshrc
-rwx--x--x   1 root     root          455 Nov  1 15:39 .profile
-rw-------   1 root     root         4014 Nov  1 15:39 .tcshrc
/sbin:
-rwx--x--x   1 root     root        23976 Nov  1 15:39 depmod
-rwx--x--x   2 root     root       274600 Nov  1 15:39 e2fsck
-rwx--x--x   1 root     root        41268 Nov  1 15:39 fdisk
-rwx--x--x   1 root     root         9396 Nov  1 15:39 fsck
-rwx--x--x   2 root     root       274600 Nov  1 15:39 fsck.ext2
-rwx--x--x   1 root     root        29556 Nov  1 15:39 getty
-rwx--x--x   1 root     root         6620 Nov  1 15:39 halt
-rwx--x--x   1 root     root        23116 Nov  1 15:39 init
-rwx--x--x   1 root     root        25612 Nov  1 15:39 insmod
-rwx--x--x   1 root     root        10368 Nov  1 15:39 kerneld
-rwx--x--x   1 root     root       110400 Nov  1 15:39 ldconfig
-rwx--x--x   1 root     root         6108 Nov  1 15:39 lsmod
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mke2fs
-rwx--x--x   1 root     root         4072 Nov  1 15:39 mkfs
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mkfs.ext2
-rwx--x--x   1 root     root         5664 Nov  1 15:39 mkswap
-rwx--x--x   1 root     root        22032 Nov  1 15:39 modprobe
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 reboot -> halt
-rwx--x--x   1 root     root         7492 Nov  1 15:39 rmmod
-rwx--x--x   1 root     root        12932 Nov  1 15:39 shutdown
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 swapoff -> swapon
-rwx--x--x   1 root     root         5124 Nov  1 15:39 swapon
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 telinit -> init
```

```
-rwx--x--x   1 root     root         6944 Nov  1 15:39 update
/tmp:
/usr:
drwx--x--x   2 root     root         1024 Nov  1 15:39 bin
drwx--x--x   2 root     root         1024 Nov  1 15:39 lib
drwx--x--x   3 root     root         1024 Nov  1 15:39 man
drwx--x--x   2 root     root         1024 Nov  1 15:39 sbin
drwx--x--x   3 root     root         1024 Nov  1 15:39 share
lrwxrwxrwx   1 root     root           10 Nov  1 15:39 tmp -> ../var/tmp
/usr/bin:
-rwx--x--x   1 root     root        37164 Nov  1 15:39 afio
-rwx--x--x   1 root     root         5044 Nov  1 15:39 chroot
-rwx--x--x   1 root     root        10656 Nov  1 15:39 cut
-rwx--x--x   1 root     root        63652 Nov  1 15:39 diff
-rwx--x--x   1 root     root        12972 Nov  1 15:39 du
-rwx--x--x   1 root     root        56552 Nov  1 15:39 find
-r-x--x--x   1 root     root         6280 Nov  1 15:39 free
-rwx--x--x   1 root     root         7680 Nov  1 15:39 head
-rwx--x--x   1 root     root         8504 Nov  1 15:39 id
-r-sr-xr-x   1 root     bin          4200 Nov  1 15:39 passwd
-rwx--x--x   1 root     root        14856 Nov  1 15:39 tail
-rwx--x--x   1 root     root        19008 Nov  1 15:39 tr
-rwx--x--x   1 root     root         7160 Nov  1 15:39 wc
-rwx--x--x   1 root     root         4412 Nov  1 15:39 whoami
/usr/lib:
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libncurses.so.4 -> libncurses.so.4.2
-rw-r-r---   1 root     root       260474 Nov  1 15:39 libncurses.so.4.2
/usr/sbin:
-r-x--x--x   1 root     root        13684 Nov  1 15:39 fuser
-rwx--x--x   1 root     root         3876 Nov  1 15:39 mklost+found
/usr/share:
drwx--x--x   4 root     root         1024 Nov  1 15:39 terminfo
/usr/share/terminfo:
drwx--x--x   2 root     root         1024 Nov  1 15:39 l
drwx--x--x   2 root     root         1024 Nov  1 15:39 v
/usr/share/terminfo/l:
-rw-------   1 root     root         1552 Nov  1 15:39 linux
-rw-------   1 root     root         1516 Nov  1 15:39 linux-m
-rw-------   1 root     root         1583 Nov  1 15:39 linux-nic
/usr/share/terminfo/v:
-rw-------   2 root     root         1143 Nov  1 15:39 vt100
-rw-------   2 root     root         1143 Nov  1 15:39 vt100-am
/var:
drwx--x--x   2 root     root         1024 Nov  1 15:39 log
drwx--x--x   2 root     root         1024 Nov  1 15:39 run
drwx--x--x   2 root     root         1024 Nov  1 15:39 tmp
/var/log:
-rw-------   1 root     root            0 Nov  1 15:39 wtmp
/var/run:
-rw-------   1 root     root            0 Nov  1 15:39 utmp
/var/tmp:
```

Exemple de contenu des répertoires d'un disque utilitaire

```
total 579
-rwxr-xr-x   1 root     root        42333 Jul 28 19:05 cpio
-rwxr-xr-x   1 root     root        32844 Aug 28 19:50 debugfs
-rwxr-xr-x   1 root     root       103560 Jul 29 21:31 elvis
-rwxr-xr-x   1 root     root        29536 Jul 28 19:04 fdisk
-rw-r-r---   1 root     root       128254 Jul 28 19:03 ftape.o
-rwxr-xr-x   1 root     root        17564 Jul 25 03:21 ftmt
-rwxr-xr-x   1 root     root        64161 Jul 29 20:47 grep
```

```
-rwxr-xr-x   1 root     root        45309 Jul 29 20:48 gzip
-rwxr-xr-x   1 root     root        23560 Jul 28 19:04 insmod
-rwxr-xr-x   1 root     root          118 Jul 28 19:04 lsmod
lrwxrwxrwx   1 root     root            5 Jul 28 19:04 mt -> mt-st
-rwxr-xr-x   1 root     root         9573 Jul 28 19:03 mt-st
lrwxrwxrwx   1 root     root            6 Jul 28 19:05 rmmod -> insmod
-rwxr-xr-x   1 root     root       104085 Jul 28 19:05 tar
lrwxrwxrwx   1 root     root            5 Jul 29 21:35 vi -> elvis
```