Disquettes d'amorce Linux : HOWTO

Tom Fawcett and Graham Chapman (traduction : Frank Pavageau, pavageau@imaginet.fr) v2.3, 4 Avril 1997

Ce document décrit la manière de créer des disquettes d'amorce, d'amorce et racine, et de maintenance. Ces disquettes peuvent être utilisées comme disquettes de secours ou pour tester de nouveaux noyaux. Note : si vous n'avez pas lu la FAQ Linux et les documents annexes tels que le HOWTO d'Installation de Linux et le Guide d'Installation de Linux, vous ne devriez pas essayer de créer de disquettes d'amorce.

Table des matières

1 Notes du traducteur

Correspondances anglais/français utilisées dans la traduction :

- boot : amorce, amorçage, amorcer, démarrage, démarrer.
- package : paquetage.
- root : racine.
- root device : périphérique racine.
- boot/root disk : disquette(s) d'amorce/racine.
- backup : archive.
- boot loader : chargeur, lanceur.
- ramdisk : disque virtuel en mémoire, disque mémoire.
- library : bibliothèque.
- dynamic library : bibliothèque partagée.

2 Introduction

2.1 Pourquoi créer des disquettes d'amorce?

Les disquettes d'amorce Linux sont utiles dans un certain nombre d'opérations, telles que :

- Tester un nouveau noyau
- Récupérer après une panne disque ou système. Une telle panne peut aller de la perte d'un secteur d'amorce à l'atterrissage d'une tête de lecture.

Il y a différents moyens d'obtenir des disquettes d'amorce :

- Utiliser celle d'une distribution comme Slackware. Cela vous permettra au moins de démarrer.
- Utiliser un utilitaire de création de disquettes de secours.
- Apprendre ce que nécessite chaque type de disquette pour fonctionner, puis créer les vôtres.

J'ai initialement choisi la dernière solution - apprendre comme cela marche pour que vous puissiez le faire vous-même. De cette manière, si quelque chose cesse de fonctionner, vous pouvez trouver comment le réparer. De plus, vous apprenez beaucoup au passage sur la manière dont Linux fonctionne.

Les utilisateurs expérimentés de Linux pourront trouver peu d'utilité à ce document. Cependant, les utilisateurs qui découvrent l'administration système Linux et qui souhaitent se protéger contre la perte de disquette racine et autres inconvenues pourront le trouver utile.

Une note sur les versions - ce document a été mis à jour pour les paquetages et versions suivants :

- Linux 2.0.6
- LILO 0.19

2.2 Appréciations et remerciements

Toute appréciation, bonne ou mauvaise, sur le contenu de ce document est bienvenue. Faîtes-nous savoir si vous trouvez des erreurs ou des oublis. Envoyez les commentaires, corrections et questions a Tom Fawcett (`fawcett@nynexst.com`) ou Graham Chapman (`grahamc@zeta.org.au`).

Nous remercions les personnes suivantes pour les corrections et les suggestions utiles à l'amélioration de ce document :

```
Randolph Bentson
```

```
Grant R. Bowman
Scott Burkett
Cameron Davidson
Bruce Elliot
Javier Ros Ganuza
HARIGUCHI Youichi
Duncan Hill
Bjxrn-Helge Mevik
Lincoln S. Peck
Dwight Spencer
Cameron Spitzer
Johannes Stille
```

2.3 Historique des modifications

v2.3, 4 Avril 1997. Modifications de cette version :

- Déplacement de la première question de la FAQ ("Pourquoi ma disquette n'amorce-t-elle pas l'ordinateur ?") dans sa propre section ("En cas de problèmes") et ajout des informations de Yard.
- Quelques changements suggérés par D. Hill et J.R. Ganuza.
- Ajout de balises de référence et de navigation à divers endroits.
- Déplacement des scripts et des ressources dans les appendices.
- Ajout d'URL pour les disquettes d'amorce des distributions, et les miroirs.

v2.2, 1er Septembre 1996. Modifications de cette version :

- Correction : donner l'attribut `ramdisk` par `rdev` sur /dev/fd0 plutôt que sur zImage.

v2.1, 18 Août 1996. Modifications de cette version :

Résumé : ce fut un nettoyage majeur pour refléter les changements entre les versions 1.2 et 2.0 du noyau. En particulier :

- Changement : remplacement des scripts shell et des listings de répertoires.
- Changement : suppression de la plupart de la réponse a la question de la FAQ sur les "disques mémoires trop grands".
- Correction : `mkfs -i` aurait du être `mke2fs -i`.
- Correction : nom du paramètre manquant dans la commande `dd` pour mettre à zéro le périphérique d'amorce.
- Correction : suppression d'un paramètre en trop de la commande `mke2fs` pour créer le système de fichier racine.

- Changement : modifications mineures pour refléter la diminution d'importance de l'utilitaire Bootkit.
- Changement : modification des références croisées des sections pour référencer le titre de la section, et non pas son numéro (un jour je rajouterais des liens hypertextes ...)
- Ajout : utilisation de `cpio` comme autre moyen de copier des fichiers.
- Ajout : astuces pour supprimer les fichiers spéciaux des périphériques inutiles.
- Ajout : question de la FAQ - que faire si rien n'arrive au moment du démarrage.
- Ajout : diverses modifications mineures.

v2.0, 12 Juin 1996. Modifications de cette version :

- Ajout : auteur et archiviste supplémentaire, Tom Fawcett.
- Ajout : section 6.3, Utilisation d'un disque mémoire.
- Ajout : section intitulée "Création de disquette de démarrage élaborée", qui explique comment utiliser avantageusement les changements des noyaux 1.3.48 et ultérieurs sur les disques mémoires.
- Changement : réécriture de la section sur le répertoire `/lib`.
- Changement : diverses astuces mineures sur le changement d'utilisation des disques mémoires.

Historique des versions :

- v1.02, 25 Juin 1995 - changements mineurs.
- v1.01, 6 Février 1995 - changement mineurs.
- v1.0, 2 Janvier 1995 - première sortie au format standard des HOWTO.
- V0.10, 1er Novembre 1994 - version originale, intitulée "Projet".

3 Disquettes

3.1 Résumé des types de disques

Je classe les disquettes ayant trait à l'amorçage dans quatre catégories. La discussion ici et à travers l'ensemble du document utilise le terme "disque" comme synonyme des disquettes, sauf indication contraire. La majorité de la discussion pourrait aussi bien s'appliquer aux disques durs.

Un résumé des types de disques et de leur utilisation est :

Amorce

Un disque contenant un noyau qui peut être chargé. Le disque peut contenir un système de fichiers et utiliser un chargeur pour démarrer, ou bien ne contenir que le noyau au début du disque. Le disque peut être utilisé pour démarrer le noyau en utilisant un système de fichiers racine sur un autre disque. Cela peut s'avérer utile si vous perdez votre chargeur en raison, par exemple, d'un essai d'installation raté.

Racine

Un disque avec système de fichiers contenant tout ce qui est nécessaire à l'utilisation d'un système Linux. Il ne contient pas nécessairement un noyau ni un chargeur.

Ce disque peut être utilisé pour faire fonctionner le système indépendemment de tout autre disque, une fois que le noyau a été chargé. Une caractéristique spéciale du noyau permet de monter un disque racine distinct une fois démarré, en copiant automatiquement le disque racine sur un disque mémoire.

Vous pouvez utiliser ce type de disque pour vérifier l'intégrité d'un autre disque sans le monter, ou pour récupérer le contenu d'un autre disque après une panne disque ou une perte de fichiers.

Amorce/racine

Un disque identique à un disque racine, mais qui contient aussi un noyau et un chargeur. Il peut être utilisé pour démarrer et faire fonctionner le système. L'avantage de ce type de disque est sa compacité - tout ce dont on a besoin est sur un seul disque. Toutefois la taille sans cesse croissante du système implique qu'il ne sera pas toujours possible de faire tout tenir sur une seule disquette, même en ayant recours à la compression.

Utilitaire

Un disque contenant un système de fichiers, mais non prévu pour être monté comme système de fichiers racine. C'est un disque de données supplémentaire. On peut utiliser ce type de disque pour transporter des utilitaires en plus quand vous avez trop à faire tenir sur votre disque racine.

Le terme "utilitaire" s'applique uniquement aux disquettes, quand vous utilisez un disque utilitaire pour stocker des programmes de récupération supplémentaires.

L'approche la plus souple pour des disquettes de secours est probablement d'utiliser des disquettes amorce et racine séparées, et une ou plusieurs disquettes utilitaires pour accueillir le supplément.

3.2 Démarrage

3.2.1 Aperçu

Tous les systèmes PC commencent leur procédure d'amorçage en exécutant du code en mémoire morte pour charger le secteur situé sur le secteur 0 du cylindre 0 du disque d'amorce et essayent de l'exécuter. Sur la plupart des disques amorçables, le secteur 0 du cylindre 0 contient :

- le code d'un chargeur comme LILO, qui trouve le noyau, le charge, et l'exécute pour commencer le véritable démarrage.
- le début d'un noyau de système d'exploitation, tel que Linux.

Si un noyau Linux a été écrit sur une disquette en accédant directement au périphérique (par `/dev/fd0` par exemple), le premier secteur sera alors le premier secteur du noyau Linux, et ce secteur continuera la procédure de démarrage en chargeant le reste du noyau et en l'exécutant. Pour une description plus détaillée du contenu du secteur de démarrage, voir la documentation fournie avec lilo-0.15 ou une version ultérieure.

Une autre manière de mettre un noyau sur un disque de démarrage est de créer un système de fichier, non pas comme racine, mais simplement pour pouvoir installer LILO et par conséquent permettre de fournir des options en ligne de commande au moment du démarrage. Par exemple, le même noyau pourrait être utilisé pour démarrer en utilisant soit un système de fichiers racine sur disque dur, soit sur disquette. Cela pourrait être utile si vous essayez de reconstruire le système de fichiers du disque dur et si vous voulez pouvoir tester les résultats en cours de route.

3.2.2 Indiquer la racine

Le noyau doit d'une façon ou d'une autre savoir quel disque et quelle partition doivent être monté comme disque racine. Cela peut se fixer de différentes manières :

- En donnant la valeur `ROOT_DEV = nom_du_périphérique` dans le makefile du noyau Linux et en recompilant le noyau (pour savoir comment recompiler le noyau, lisez la FAQ Linux et regardez dans `/usr/src/linux`). Des commentaires dans le makefile Linux décrivent les valeurs autorisées pour `nom_du_périphérique`.
- En utilisant l'utilitaire `rdev` :

  ```
  rdev nom_du_fichier nom_du_peripherique
  ```

  Cela change le périphérique racine du noyau contenu dans `nom_du_fichier` pour être `nom_du_périphérique`. Par exemple :

  ```
  rdev zImage /dev/sda1
  ```

  Le noyau contenu dans zImage a maintenant comme périphérique racine la première partition du premier disque SCSI.

Il y a d'autres manières d'utiliser la commande `rdev`. Essayez :

```
rdev -h
```

et vous obtiendrez les options de la commande.

Il n'y a en général pas besoin de configurer le périphérique racine pour une disquette de boot, car le noyau actuellement utilisé pour démarrer a déjà la valeur du disque racine. Le besoin peut se faire sentir, par contre, si vous récupérez le noyau d'une autre machine, par exemple celui d'une distribution, ou si vous voulez utiliser le noyau pour démarrer avec une disquette racine. Vérifier la valeur du disque racine actuel est probablement une bonne idée, au cas où il serait erroné. Pour vérifier avec `rdev` le périphérique racine actuel d'un fichier noyau, tapez la commande :

```
rdev <nom_fichier>
```

Il est possible de changer le périphérique racine d'un noyau par d'autres moyens que `rdev`. Pour plus de détails, voir la FAQ à la fin de ce document.

3.2.3 Copier le noyau sur la disquette d'amorce

Une fois que le noyau a été configuré, il doit être copié sur la disquette d'amorce.

Les commandes décrites ci-dessous (et à travers le reste du HOWTO) supposent que les disquettes ont été formattées. Si ça n'est pas le cas, utilisez `fdformat` pour formatter les disquettes avant de continuer.

Si le disque ne doit pas contenir de système de fichiers, alors le noyau peut être copié en utilisant la commande `dd`, comme suit :

```
dd if=fichier_origine of=peripherique

ou      fichier_origine est le nom du noyau
et      peripherique est l'interface d'acces a la couche
        physique de la disquette, generalement /dev/fd0
```

La commande `cp` peut aussi être utilisée :

```
cp nom_du_fichier nom_du_peripherique
```

Par exemple :

```
        dd if=zImage of=/dev/fd0
ou
        cp zImage /dev/fd0
```

Le paramètre `seek` de la commande `dd` ne doit PAS être utilisé. Le fichier doit être copié au début du secteur d'amorce (secteur 0, cylindre 0), et ne pas préciser de paramètre `seek` permet cela.

Le nom du périphérique à utiliser est généralement `/dev/fd0` pour le lecteur de disquette principal (c'est à dire le lecteur "A:" sous DOS), et `/dev/fd1` pour le secondaire. Ces noms de périphériques permettront au noyau de détecter automatiquement les caractéristiques des lecteurs. Ces caractéristiques peuvent être précisée au noyau en utilisant d'autres noms de périphériques : par exemple, `/dev/fd0H1440` indique un lecteur haute densité 1.44 Mo. Il est rare d'avoir à recourir à ces noms de périphériques spécifiques.

Si le noyau doit être copié sur un disque d'amorce contenant un système de fichiers, le disque est alors monté sur un point d'entrée adéquat sur un système de fichiers actuellement monté, puis la commande `cp` est utilisée. Par exemple :

```
mount -t ext2 /dev/fd0 /mnt
cp zImage /mnt
umount /mnt
```

Notons que pour pratiquement toutes les opérations dans ce HOWTO, l'utilisateur doit agir en tant que super-utilisateur.

3.3 Racine

3.3.1 Aperçu

Un disque racine contient un système fonctionnel Linux complet, mais sans nécessairement inclure un noyau. En d'autres termes, le disque peut ne pas être amorçable, mais une fois que le noyau fonctionne, le disque racine contient tout ce qui est nécessaire au fonctionnement d'un système Linux complet. Pour pouvoir faire ceci, le disque doit contenir le minimum pour un système Linux :

- Système de fichiers.
- Ensemble minimum de répertoires - `dev`, `proc`, `bin`, `etc`, `lib`, `usr`, `tmp`.
- Ensemble d'utilitaires basiques - `bash` (pour avoir un shell), `ls`, `cp`, etc...
- Ensemble de fichiers de configuration minimal - `rc`, `inittab`, `fstab`, etc...
- Bibliothèque pour fournir les fonctions de base nécessaires par les utilitaires.

Bien sûr, tout système devient utile lorsque que l'on peut faire fonctionner quelque chose dessus, et une disquette racine ne devient en général utile que si vous pouvez avoir des actions du type :

- Vérifier un système de fichiers sur un autre disque ; par exemple, pour vérifier votre système de fichiers racine sur votre disque dur, vous devez pouvoir démarrer Linux depuis un autre disque, comme il est possible avec une disquette racine. Vous pouvez alors lancer `fsck` sur votre disque racine habituel puisqu'il n'est pas monté.
- Récupérer tout ou partie de votre disque racine habituel depuis une sauvegarde en utilisant des utilitaires d'archivage et de compression tels que `cpio`, `tar`, `gzip` et `ftape`.

3.4 Amorce/racine

Pour qu'une disquette avec un système de fichier racine soit efficace, vous devez pouvoir le faire fonctionner depuis un disque mémoire, c'est-à-dire un disque émulé en mémoire vive. Cela évite de faire fonctionner le système à une vitesse d'escargot, ce qui est le cas avec une disquette. Le HOWTO Ftape indique qu'un disque mémoire sera nécessaire lorsque l'on utilise Ftape car Ftape nécessite l'usage exclusif du controlleur de disquette.

Il y a un autre bénéfice du fait d'utiliser un disque mémoire - le noyau Linux inclut la possibilité d'avoir un disque mémoire automatique, où il copiera automatiquement, dans certaines circonstances, le contenu d'une disquette racine, et qu'il utilisera alors comme disque racine au lieu de la disquette. Trois points positifs en résultent :

- Le système fonctionne bien plus vite.
- Le lecteur de disquette est libéré automatiquement pour pouvoir avoir accès à d'autres disquettes sur un système mono-lecteur.

- Avec de la compression, la taille de l'image d'un disque mémoire sur un disque peut être réduit de manière significative (en general, 40% de la taille) par rapport à la taille du disque mémoire lui-même. Cela signifie qu'une disquette 1.44 Mo peut contenir une racine d'environ 3 Mo.

Pour les noyaux 1.3.48 et ultérieurs, le code du disque mémoire a été substantiellement réécrit. Vous avez des options supplémentaires et les commandes pour utiliser le disque mémoire sont quelque peu différentes. La section 5 discute de cela plus en détail.

Vous devez configurer votre noyau pour avoir le support disque mémoire (ramdisk), mais le disque mémoire pouvant être agrandi de manière dynamique, il n'est pas nécessaire d'en préciser la taille. `rdev -r` n'est plus utilisé pour préciser la taille du disque mémoire, mais change par contre l'option disque mémoire dans l'image du noyau. La section 5 discute de cela de manière plus détaillée.

Si vous avez un noyau *antérieur* à 1.3.48, les points suivants sont nécessaires. Notez que cela s'applique *UNIQUEMENT* au noyaux antérieurs à la version 1.3.48.

- Le système de fichiers sur la disquette doit être soit un système *minix*, soit un système *ext2*. Le système *ext2* est en général préféré. Notez que si vous avez un noyau Linux antérieur à 1.1.73, vous devriez voir les commentaires de la section intitulée “Systèmes de fichiers” pour savoir si votre noyau supportera *ext2*. Si votre noyau est ancien, vous pourriez avoir à utiliser *minix*. Cela ne pose pas de problèmes particuliers.
- Le noyau doit être configuré pour avoir un disque mémoire, et il doit être au moins aussi gros que la taille de la disquette.

On peut configurer le noyau pour avoir un disque mémoire de diverses manières :

- En décommentant la macro `RAMDISK` dans le makefile du noyau Linux, pour qu'elle indique :

  ```
  RAMDISK = -DRAMDISK=1440
  ```

  pour définir un disque mémoire de 1440 blocs de 1K, la taille d'une disquette haute densité.
- En utilisant l'utilitaire `rdev`, disponible sur la plupart des systèmes Linux. Cet utilitaire affiche ou change les valeurs de différentes choses dans le noyau, dont la taille désirée pour un disque mémoire. Pour configurer le noyau afin d'avoir un disque mémoire de 1440 blocs, dans un fichier appelé `zImage`, tapez :

  ```
  rdev -r zImage 1440
  ```

  cela pourrait changer dans le futur, bien sûr. Pour savoir ce que votre version de `rdev` fait, tapez la commande :

  ```
  rdev -h
  ```

  et il devrait afficher ses options.
- En utilisant le paquetage du chargeur LILO pour configurer votre noyau au moment du démarrage. Cela peut se faire en utilisant le paramètre de configuration de LILO :

  ```
  ramdisk = 1440
  ```

pour demander un disque mémoire de 1440 blocs de 1K au moment du boot.

- En interrompant un démarrage automatique de LILO et en ajoutant `ramdisk=1440` sur la ligne de commande. Par exemple, une telle commande pourrait être :

```
zImage ramdisk=1440
```

Voir la section sur LILO pour plus de détails.

- En éditant le fichier du noyau et en modifiant les valeurs au début du fichier qui stockent la taille du disque mémoire. C'est vraiment la dernière chose à faire, mais c'est possible. Voir la FAQ à la fin du document pour plus de détails.

Le moyen le plus simple est la configuration de LILO, car vous aurez besoin de créer une configuration pour LILO de toute manière. Pourquoi ne pas y ajouter la taille du disque mémoire ?

La configuration de LILO est brièvement décrite dans la section intitulée “LILO” ci-dessous, mais il est recommandé d'obtenir la dernière version stable en cours de LILO depuis votre miroir Linux le plus proche, et de lire la documentation l'accompagnant.

Les disques mémoires peuvent avoir une taille plus grande que celle d'une disquette, et peuvent contenir un système de fichiers aussi gros que leur taille. Cela peut être utile pour charger tous les programmes nécessaires à la réparation sur un seul disque mémoire performant. La méthode pour faire cela est décrite dans la FAQ, dans la réponse à la question “Comment puis-je créer un grand système de fichier sur disque mémoire ?”

3.5 Utilitaire

Souvent, un seul disque n'est pas suffisant pour contenir l'ensemble des programmes dont vous avez besoin pour pouvoir analyser, réparer et restaurer des disques corrompus. Quand vous en arrivez à inclure `tar`, `gzip`, `e2fsck`, `fdisk`, `Ftape`, etc... ils prennent assez de place pour une nouvelle disquette, voire plus si vous avez besoin de beaucoup d'outils.

Cela entraîne qu'un kit de secours nécessite souvent une disquette utilitaire, avec un système de fichiers contenant tous les extras dont vous avez besoin. Ce système de fichiers peut alors être monté à un endroit pratique, comme `/usr`, sur le système amorce/racine.

Créer un système de fichiers est relativement simple, et il est décrit dans la section intitulée “Systèmes de fichiers”.

4 Composants

4.1 Systèmes de fichiers

Le noyau Linux supporte maintenant deux types de systèmes de fichiers pour les disques racines à copier automatiquement sur disque mémoire. Ce sont *minix* et *ext2*, *ext2* étant le système de fichier préféré. Ce support *ext2* a été rajouté quelque part entre les versions 1.1.17 et 1.1.57, je ne sais pas exactement quand. Si vous avez un noyau parmi ces versions, éditez `/usr/src/linux/drivers/block/ramdisk.c` et cherchez le mot “ext2”. Si vous ne le trouvez pas, vous aurez à utiliser un système de fichiers *minix*, créé au moyen de la commande `mkfs`. Si vous utilisez *ext2*, vous pourrez trouver pratique d'utiliser l'option `-i` pour allouer plus d'i-noeuds que par défaut ; on recommande `-i 2000` pour ne pas manquer d'i-noeuds. Sinon, vous pouvez économiser des i-noeuds en supprimant les nombreux fichiers inutiles de `/dev`. `Mke2fs` créera par défaut 360 i-noeuds sur une disquette 1.44 Mo. Pour ma part, 120 i-noeuds sont suffisants sur mon actuelle disquette de secours racine, mais si vous incluez tous les périphériques dans le répertoire `/dev`, vous dépasserez allègrement 360. Utiliser un système de fichiers compressé pour la racine permet un système plus grand, et donc plus d'i-noeuds par défaut, mais vous pourriez cependant devoir réduire le nombre de fichiers ou bien augmenter le nombre d'i-noeuds.

Pour créer un système de fichiers *ext2* sur une disquette, sur ma machine, je lance la commande suivante :

```
mke2fs -m 0 /dev/fd0
```

La commande `mke2fs` va automatiquement détecter l'espace disponible et se configurer en fonction. Eventuellement, la taille de la disquette en blocs de 1Ko peut être spécifiée pour accélérer l'opération de `mke2fs`. Le paramètre `-m 0` évite à `mke2fs` de réserver de la place pour root, ce qui laisse plus de place disponible sur le disque.

Une manière simple de tester le résultat est de créer un système en utilisant la commande ci-dessus ou une commande approchant, et d'essayer ensuite de monter la disquette. S'il s'agit d'un système *ext2*, alors la commande :

```
mount -t ext2 /dev/fd0 /<point d'entree>
```

devrait fonctionner.

4.2 Noyau

4.2.1 Construire un noyau personnalisé

Dans la plupart des cas, il serait possible de copier votre noyau actuel et d'amorcer la disquette avec ça. Cependant, il peut y avoir des cas où vous souhaitez construire un noyau différent.

Une raison peut être la taille. Le noyau est un des plus gros fichiers, sur un système de taille minimale, et si vous souhaitez créer une disquette amorce/racine, vous aurez à réduire la taille du noyau autant que possible. Le noyau supporte désormais le changement de disquette après amorçage et avant de monter la racine, et comme il n'est donc plus nécessaire d'entasser le noyau sur le même disque que tout le reste, ces commentaires ne s'appliquent que si vous décidez de construire une disquette amorce/racine.

Il y a deux manières de réduire la taille du noyau :

- Le construire avec le minimum d'options nécessaires pour faire fonctionner le système désiré. Cela impose de ne conserver que ce dont vous avez besoin. Le support réseau est un candidat à l'élimination, tout comme le support pour tous disques et autres périphériques que vous n'utilisez pas avec votre système amorce/racine.

– Le compresser, en utilisant l'option standard de compression du noyau incluse dans le makefile :

```
make zImage
```

Reportez vous à la documentation incluse avec les sources du noyau pour une information à jour sur la construction de noyaux compressés. Notez que les sources du noyau sont généralement dans `/usr/src/linux`.

Une fois choisi le minimum d'options à inclure dans un noyau, vous devez choisir quoi remettre dedans. L'utilisation la plus commune d'une disquette amorce/racine est probablement d'examiner et restaurer un système de fichiers racine corrompu, et cela nécessite éventuellement du support dans le noyau.

Par exemple, si vos sauvegardes sont stockées sur bandes en utilisant `Ftape` pour accéder au lecteur et que vous perdez votre disque racine ainsi que les disques contenant `Ftape`, vous ne pourrez alors pas restaurer votre système à partir des bandes de sauvegarde. Vous devrez réinstaller Linux, récupérer et réinstaller `Ftape`, et seulement alors essayer de relire vos sauvegardes.

Il est souhaitable de maintenir une copie des mêmes versions des utilitaires servant à écrire les sauvegardes, afin de ne pas perdre de temps à essayer d'installer des versions ne pouvant pas lire vos bandes de sauvegarde.

Ce qu'il faut comprendre, c'est que tout support d'entrées/sorties que vous avez ajouté à votre noyau pour réaliser des sauvegardes doit aussi être ajouté à votre noyau d'amorce/racine.

La procédure pour reconstruire le noyau est decrite dans la documentation accompagnant le noyau. C'est assez simple à suivre, donc commencez par regarder dans `/usr/src/linux`. Si vous avez des problèmes pour construire un noyau, vous ne devriez probablement pas essayer de construire des disques amorce/racine de toute manière.

4.3 Périphériques

Tout système Linux nécessite un répertoire `/dev` contenant un fichier spécial pour tous les périphériques utilisées par le système. Le répertoire lui-même est un répertoire normal, et peut être créé avec la commande `mkdir` de la manière habituelle. Les fichiers spéciaux de périphériques doivent par contre être créés d'une certaine manière, avec la commande `mknod`.

Il y a un raccourci cela dit - copiez votre répertoire `/dev` existant, et supprimez ceux dont vous n'avez pas besoin. La seule chose à faire est de copier les fichiers spéciaux de périphériques en utilisant l'option `-R`.

> Attention : la commande `cp` fournie avec la plus récente version des fileutils (3.13) semble ne pas respecter l'option `-R`.

Cela copie le répertoire sans essayer de copier le contenu des fichiers. Si vous utilisez un `r` minuscule, à savoir "`-r`", le comportement sera très différent, puisque vous essaierez de copier tout le contenu de tous vos disques - ou tout au moins ce qui tiendra sur une disquette ! Faites donc attention, et utilisez la commande :

```
cp -dpR /dev /mnt
```

en supposant que la disquette est montée sur `/mnt`. Les options `dp` conservent les liens symboliques en tant que tels lors de la copie (au lieu de copier le fichier pointé) ainsi que les attributs des fichiers originaux, préservant donc les informations de propriété.

Vous pouvez aussi utiliser l'option `-p` de `cpio`, car `cpio` traitera les fichiers spéciaux de périphériques correctement et n'essaiera pas d'en copier le contenu. Par exemple :

```
cd /dev
find . -print | cpio -pmd /mnt/dev
```

copiera tous les fichiers spéciaux de `/dev` sur `/mnt/dev`. En fait, cela copiera tous les fichiers dans l'arborescence de répertoires commençant à `/dev`, et créera tous les sous-répertoires nécessaires dans l'arborescence cible.

Si vous cherchez la complexité, utilisez `ls -l` pour connaître les numéros de périphériques majeurs et mineurs de ceux que vous voulez, et créez les sur la disquette en utilisant `mknod`.

De nombreuses distributions incluent un script shell appelé `MAKEDEV` dans le répertoire `/dev`. Ce script shell pourrait être utilisé les périphériques, mais il est sans doute plus facile de copier ceux existant, surtout pour un disque de secours.

Quelle que soit la manière dont le répertoire de périphériques est copié, cela vaut la peine de vérifier que tous les fichiers spéciaux dont vous avez besoin sont présents sur la disquette de secours. Par exemple, `Ftape` utilise des périphériques de bande, et vous devrez donc copier tous ceux-là.

Notez qu'un i-noeud est requis pour chaque fichier spécial et que les i-noeuds peuvent parfois être rares, surtout pour des systèmes de fichiers sur disquette. Il est donc utile de supprimer tous les fichiers spéciaux dont vous n'avez pas besoin dans le répertoire `/dev` de la disquette. De nombreux périphériques sont clairement inutiles sur des systèmes spécifiques. Par exemple, si vous n'avez pas de disques SCSI, vous pouvez supprimer sans problèmes tous les fichiers de périphériques commençant par "`sd`". De même, si vous n'avez pas l'intention d'utiliser le port série, tous les fichiers commençant par "`cua`" peuvent disparaître.

4.4 Répertoires

Il est peut-être possible de faire fonctionner un système Linux uniquement avec `/dev`, `/proc` et `/etc`. Je ne sais pas - je n'ai jamais essayé. Toutefois, cela risque de s'avérer difficile, car sans bibliothèques partagées tous vos exécutables devraient être compilés statiquement. Il est raisonnable d'avoir au minimum les répertoires :

/dev

Nécessaire pour les entrées/sorties avec les périphériques

/proc

Nécessaire pour la commande ps

/etc

Fichiers de configuration du système

/bin

Exécutables des utilitaires considérés comme faisant partie du système

/lib

Bibliothèques partagées nécessaires au fonctionnement

/mnt

Un point de montage pour effectuer des opérations de maintenance sur d'autres disques

/usr

Utilitaires et applications supplémentaires

L'arborescence présentée ici n'est à utiliser que sur une disquette racine. Consultez le Standard de Système de Fichiers Linux (Linux File System Standart) pour de plus amples informations sur la structure des systèmes de fichiers sur des systèmes Linux "standards".

Quatre de ces répertoires peuvent être créés facilement :

- `/dev` est décrit ci-dessus dans la section intitulée Périphériques.
- `/proc` doit juste être présent. Il n'y a juste qu'à créer le répertoire avec `mkdir`.
- `/mnt` et `/usr` sont uniquement inclus dans cette liste comme points de montage à utiliser une fois que le système amorce/racine fonctionne. Ici aussi, ces répertoires n'ont qu'à être créés.

Les 3 répertoires restants sont décrits dans les sections suivantes.

4.4.1 /etc

Ce répertoire doit contenir un certain nombre de fichiers de configuration. Sur la plupart des systèmes, on peut les répartir en 3 groupes :

- Nécessaire en permanence, par exemple `rc`, `fstab`, `passwd`.
- Eventuellement nécessaire, mais personne n'est sûr.
- Le bazar qui s'est accumulé

Les fichiers non essentiels peuvent être identifiés avec la commande :

```
ls -ltru
```

On obtient la liste des fichiers dans l'ordre inverse de leur date de dernier lecture, et si des fichiers ne sont jamais lus, ils peuvent être omis de la disquette racine.

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

- Ceux que je dois configurer pour un système amorce/racine :

```
rc.d/*  scripts de demarrage et de changement de niveau de
        fonctionnement (run level)
fstab   liste de systemes de fichiers a monter
inittab parametres pour le processus init_- le premier
        processus lance au demarrage.
```

- Ceux que je dois nettoyer pour un système amorce/racine :

```
passwd  liste des utilisateurs
shadow  qui contient les mots de passe
```

  Ces fichiers doivent être nettoyés sur des systèmes sécurisés afin de ne pas copier les mots de passe des utilisateurs ailleurs que sur le système, et pour que lorsque vous démarrez depuis une disquette, les logins invalides soient rejetés.

  > Notez qu'il y a une raison pour *ne pas* nettoyer `passwd` et `shadow`. `Tar` (et probablement d'autres programmes d'archives) stocke les noms d'utilisateurs et de groupes avec les fichiers. Si vous restaurez des fichiers sur votre disque dur depuis une bande, les fichiers seront restaurés avec leurs noms originaux, Si ces noms n'existent pas dans `passwd/group` quand ils sont restaurés, les IUD/GID ne seront pas corrects.

- Le reste. Ils fonctionnent pour l'instant, je n'y touche donc pas.

Parmi tout ça, je n'ai réellement que deux fichiers à configurer, et leur contenu est étonnamment petit.

- `rc` doit contenir :

```
#!/bin/sh
/etc/mount -av
/bin/hostname boot_root
```

  où je n'ai pas vraiment besoin de lancer `hostname` - ça a juste meilleure allure si je le fais. Même `mount` est uniquement nécessaire pour monter `/proc` pour que `ps` fonctionne - Linux fonctionne sans ça, bien que les opérations de secours soient assez limitées sans `mount` !

- `fstab` doit contenir :

```
/dev/ram        /               ext2    defaults
/dev/fd0        /               ext2    defaults
/proc           /proc           proc    defaults
```

  Je ne crois pas que la première ligne soit vraiment nécessaire, mais j'ai remarqué que si je ne la mets pas, `mount` ne monte pas `/proc`.

`Inittab` devrait convenir tel quel, sauf si vous voulez vous assurer que les utilisateurs par port série ne peuvent pas se connecter. Pour éviter cela, commentez toutes les lignes avec `/etc/getty` contenant un périphérique `tty` ou `ttyS` au bout de la ligne. Laissez les ports `tty` pour pouvoir vous connecter sur la console.

`Inittab` définit ce que le système lance ou relance à différents moments, comme le démarrage, le passage en mode multi-utilisateurs, les pertes d'alimentation, etc. Il faut faire s'assurer que les commandes présentes dans `inittab` correspondent à des programmes présents, ainsi qu'au bon répertoire. Si vous placez vos fichiers de commandes sur le disque de secours en utilisant l'exemple de contenu de répertoires fourni dans ce HOWTO comme guide, puis copiez votre `inittab` sur le disque de secours sans le vérifier, il y a de fortes chances que cela ne fonctionne pas, car la moitié des lignes de l'`inittab` feront référence à des programmes inexistants ou placés dans le mauvais répertoire.

Il faut aussi noter que certains programmes ne peuvent être déplacés d'un répertoire vers un autre sans mal fonctionner au démarrage car ils référencent directement dans leur code le nom d'un autre programme qu'ils essayent de lancer. Par exemple sur mon système, `/etc/shutdown` contient directement le chemin `/etc/reboot`. Si je déplace `reboot` vers `/bin/reboot` puis que je lance une commande `shutdown`, elle ne va pas marcher puisqu'elle ne pourra pas trouver le fichier `reboot`.

Pour le reste, copiez simplement tous les fichiers texte dans votre répertoire `/etc`, ainsi que tous les exécutables dans votre répertoire `/etc` dont vous n'êtes pas sûr de pouvoir vous débarrasser. Pour vous guider, consultez la sortie exemple de `ls` dans "Contenu des répertoires du disque racine (`ls -lR`)" - c'est ce que j'ai, et ça devrait donc vous suffire si vous copiez ces fichiers - mais notez que les systèmes peuvent être très différents, et on ne peut donc pas être certain qu'un ensemble de fichiers sur votre système est équivalent au même sur le mien. La seule méthode sure est de démarrer avec inittab et déterminer ce qui est nécessaire.

La plupart des systèmes utilisent maintenant un répertoire `/etc/rc.d` contenant des scripts shell pour les différents niveaux de fonctionnement (run levels). Le minimum absolu est un fichier `rc` unique, mais il sera sans doute beaucoup plus simple en pratique de copier le fichier `inittab` et le répertoire `/etc/rc.d` de votre système, et nettoyer les scripts shell dans `rc.d` pour supprimer tous les traitements inutiles à un environnement système sur disquette.

4.4.2 /bin

C'est le bon endroit pour placer les autres utilitaires dont vous avez besoin pour les opérations de base, tels que `ls`, `mv`, `cat`, `dd`, etc.

Voir la section intitulée "Exemple de `ls -lR` pour amorce/racine" pour une liste des fichiers que je place dans le répertoire `/bin` de ma disquette amorce/racine. Vous pourrez remarquer qu'aucun des utilitaires nécessaires à la restauration d'une sauvegarde, tels que `cpio`, `tar`, `gzip`, etc, ne s'y trouvent. C'est parce que je les place sur une disquette utilitaire séparée, pour garder de la place sur la disquette amorce/racine. Après avoir démarré sur ma disquette amorce/racine, son contenu est copié dans le disque mémoire, laissant le lecteur de disquette libre pour monter une autre disquette, la disquette utilitaire. Je la monte en général sur `/usr`.

La création d'une disquette utilitaire est décrite ci-dessous dans la section intitulée "Ajouter des disquettes utilitaires".

4.4.3 /lib

Dans `/lib` vous placez certaines bibliothèques partagées ainsi que les chargeurs. Si les bibliothèques nécessaires ne sont pas trouvées dans votre répertoire `/lib`, le système ne pourra pas démarrer. Si vous avez de la chance, vous verrez un message d'erreur vous expliquant pourquoi.

Pratiquement tous les programmes nécessitent au moins la bibliothèque `libc` :

```
libc.so.X
```

où X représente le numéro de version courante. Vérifiez dans votre répertoire `/lib`. Notez que `libc.so.4` peut être un lien symbolique vers une bibliothèque `libc` avec un numéro de version dans le nom. Si vous lancez la commande :

```
ls -l /lib
```

vous obtiendrez quelque chose du genre :

```
libc.so.4 -> libc.so.4.5.21
```

Dans ce cas, la bibliothèque `libc` dont vous avez besoin est `libc.so.4.5.21`. Ce n'est qu'un exemple - la bibliothèque `libc` *ELF* est actuellement `libc.so.5.xxxx`.

Pour trouver d'autres bibliothèques, vous devrez tester les dépendances de tous les binaires que vous souhaitez utiliser. La commande `ldd` sert à ça. Par exemple, sur mon système la commande :

```
ldd /bin/mount
```

renvoie comme résultat :

```
/bin/mount:
        libc.so.5 => /lib/libc.so.5.2.18
```

indiquant que `/bin/mount` a besoin de la bibliothèque `libc.so.5`, qui est un lien symbolique sur `libc.so.5.2.18`.

Vous devez aussi placer dans `/lib` un ou plusieurs chargeurs pour charger les bibliothèques. Le fichier du chargeur est soit `ld.so` (pour les bibliothèques au format *a.out*), soit `ld-linux.so` (pour les bibliothèques *ELF*). Si vous n'êtes pas sûr duquel vous avez besoin, lancez la commande “`file`” sur la bibliothèque. Par exemple, sur mon système :

```
file /lib/libc.so.5.2.18
```

me répond :

```
/lib/libc.so.5.2.18: ELF 32-bit LSB shared object ...
```

et par conséquent elle a besoin d'un chargeur *ELF*. Si vous avez une bibliothèque *a.out*, vous obtiendrez un résultat du genre :

```
/lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC) ...
```

Copiez le(s) chargeur(s) spécifique(s) dont vous avez besoin.

Il faut vérifier que les bibliothèques et chargeurs correspondent bien aux binaires utilisés. Si le noyau ne peut pas charger une bibliothèque indispensable, il plantera en général sans message d'erreur.

4.5 LILO

4.5.1 Aperçu

Pour que l'amorce/racine soit utilisable, elle doit être amorçable. Pour ce faire, le plus simple est d'installer un chargeur, qui est un morceau de code executable stocké sur le secteur 0 du cylindre 0 de la disquette. Voir la section ci-dessus intitulée "Disquettes de boot" pour une description du processus.

LILO est un chargeur longuement testé et éprouvé disponible sur tout site miroir Linux. Il vous permet de configurer le chargeur, en particulier :

- Quel périphérique doit être utilisé comme disque racine
- S'il faut utiliser un disque mémoire

4.5.2 Exemple de configuration de LILO

Cela permet de spécifier au noyau comment il doit démarrer. Mon fichier de configuration de LILO pour l'amorce/racine, utilisé avec LILO 0.15, est :

```
boot = /dev/fd0
install = ./mnt/boot.b
map = ./mnt/lilo.map
delay = 50
message = ./mnt/lilo.msg
timeout = 150
compact
image = ./mnt/zImage
        ramdisk = 1440
        root = /dev/fd0
```

Notez que je ne l'ai pas testé récemment, car je n'utilise plus de disquettes amorce/racine utilisant LILO. Il n'y a aucune raison de supposer que cela ne fonctionne plus, mais si vous essayez et que cela ne marche pas, vous devez lire la documentation de LILO pour savoir pourquoi.

Notez aussi que les systèmes amorce/racine n'ont plus besoin de LILO, car depuis la version 1.3.48, le noyau supporte le chargement d'un système de fichiers compressé depuis la même disquette que le noyau. Voire la section 5 pour les détails.

Si vous avez un noyau postérieur à la version 1.3.48, la ligne "`ramdisk = 1440`" est inutile et doit être retirée.

Notez que `boot.b`, `lilo.msg` et le noyau doivent préalablement avoir été copiés sur la disquette en utilisant une commande du genre :

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

Si cela n'est pas fait, LILO ne fonctionnera pas correctement au démarrage si le disque dur n'est pas disponible, et un disque de secours qui nécessite la présence du disque dur pour démarrer est d'une utilité douteuse.

Je lance `lilo` avec la commande :

```
/sbin/lilo -C <fichier de configuration>
```

Je le lance depuis le répertoire contenant le répertoire `mnt` où la disquette est montée. Cela signifie que je demande à LILO d'installer un chargeur sur le périphérique de démarrage (`/dev/fd0` dans ce cas), pour démarrer un noyau situé dans le répertoire racine de la disquette.

J'ai aussi spécifié que je veux utiliser la disquette comme périphérique racine, et que je veux qu'un disque mémoire de 1440 blocs de 1 Ko soit créé, à savoir la taille de la disquette. Puisque j'ai créé un système de fichiers *ext2* sur la disquette, toutes les conditions sont remplies pour que Linux utilise automatiquement le disque mémoire comme périphérique racine, et y copie le contenu de la disquette.

Les possibilités de Linux concernant les disques mémoires sont décrites de manière plus poussées dans la section ci-dessus intitulée "Disques mémoires et systèmes amorce/racine".

On peut aussi envisager d'utiliser le paramètre "`single`" pour démarrer Linux en mode mono-utilisateur. Cela permet d'empêcher les utilisateurs de se connecter par les ports série.

J'utilise aussi les instructions "`DELAY`", "`MESSAGE`" et "`TIMEOUT`" pour que lorsque je démarre le disque, LILO me laisse la possibilité d'entrer des options en ligne de commande si je le souhaite. Je n'en ai pas besoin actuellement, mais on ne sait jamais si l'on ne souhaitera pas changer de périphérique racine ou monter un système de fichiers en lecture seule.

Le fichier de messages que j'utilise contient le message :

```
Disquette amorce/racine Linux
=============================

Entrez une ligne de commandes de la forme_:

        zImage [ options en ligne ]

Si rien n'est entre, Linux sera charge avec les parametres
par defaut dans 15 secondes.
```

Il ne s'agit que d'un aide-mémoire pour me rappeler mes options.

Les lecteurs sont invités à lire la documentation de LILO de manière attentive avant d'essayer d'installer quoi que ce soit. Il est assez facile de détruire des partitions si vous utilisez le mauvais paramètre "`boot =` ". Si vous manquez d'expérience, NE LANCEZ PAS LILO avant d'être sûr que vous le comprenez et d'avoir vérifié de nombreuses fois vos paramètres.

Notez que vous devez relancer LILO à chaque fois que vous changez le noyau, pour que LILO puisse mettre à jour sa cartographie en fonction du nouveau noyau. Il est en fait possible de remplacer le noyau par un autre quasiment identique sans relancer LILO, mais mieux vaut ne pas parier la dessus - si vous changez de noyau, relancez LILO.

4.5.3 Retirer LILO

Je peux aussi ajouter ceci tant que j'en suis à parler de LILO : si vous installez par erreur LILO sur un disque contenant DOS, vous pouvez toujours remplacer le secteur d'amorce par le chargeur du DOS en tapant la commande DOS :

```
FDISK /MBR
```

où MBR signifie “Master Boot Record”. Notez que quelques puristes ne sont pas d'accord, éventuellement avec raison, mais cela marche.

4.5.4 Options utiles de LILO

LILO a plusieurs options utiles qu'il est bon de se rappeler lorsque l'on crée des disques de démarrage :

- Options en ligne de commande - vous pouvez entrer des options en ligne de commande pour changer le périphérique racine, la taille du disque mémoire (pour les noyaux antérieurs à 1.3.48), des paramètres spéciaux pour certains périphériques, ou d'autres choses encore. Si vous incluez l'instruction “`DELAY = nn`” dans votre fichier de configuration LILO, LILO fera alors une pause pour vous permettre de sélectionner une image noyau à amorcer, et d'entrer sur la même ligne, des options. Par exemple :

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

  passera les paramètres *aha152x* jusqu'au gestionnaire de disque SCSI `aha152x` (à condition que ce gestionnaire ait été inclus à la construction du noyau) et demandera à ce que le système de fichiers racine soit monté en lecture seule.

- Option en ligne de commande “`lock`” - cette option demande à LILO de stocker la ligne de commande entrée comme ligne de commande par défaut pour tous les démarrages futurs. C'est particulièrement utile lorsque vous utilisez un périphérique qui ne peut pas être auto-sélectionné. En utilisant “`lock`” vous évitez d'avoir à taper la chaîne de paramètre du gestionnaire à chaque démarrage. Par exemple :

  ```
  zImage aha152x=0x340,11,3,1 root=/dev/sda8 ro lock
  ```

- instruction de configuration `APPEND` - cela permet de stocker des chaînes de paramètres pour gestionnaires dans la configuration, comme alternative à l'utilisation de l'option “`lock`”. Notez que tous les mots clés de la forme *mot=valeur* DOIVENT être entourés par des guillemets. Par exemple :

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

- instruction de configuration `DELAY` - cela provoque une pause de `DELAY` dixièmes de secondes et permet à l'utilisateur d'interrompre le démarrage automatique de la ligne de commande par défaut, afin qu'il puisse en entrer une autre.

5 Création de disquette de démarrage élaborée

5.1 Aperçu

Les sections précédentes de ce document couvrent les bases de la création de disquettes amorce et racine, et s'appliquent à quasiment tous les noyaux jusqu'aux versions actuelles (2.0, le dernier noyau stable).

Les noyaux 1.3.48 et ultérieurs contiennent une réécriture substantielle du code du disque mémoire, rajoutant d'importantes fonctionnalités nouvelles. Ces noyaux peuvent automatiquement détecter de systèmes de fichiers compressés, les décompresser et les charger dans le disque mémoire au démarrage. Les systèmes de fichiers racine peuvent être placés sur un second disque, et à partir du noyau 1.3.98 environ, les disques mémoires peuvent être agrandis dynamiquement.

Ensemble, ces nouvelles fonctionnalités permettent aux disques d'amorce de contenir substantiellement plus de données qu'ils ne pouvaient le faire jusqu'alors. Avec de la compression, un disque de 1722 Ko contient désormais jusqu'à 3.5 Mo de fichiers. Comme tous ceux qui ont créés des disques d'amorce le savent, on passe beaucoup de temps à élaguer l'ensemble des fichiers et à trouver des versions réduites des fichiers pour que le tout tienne sur un petit système de fichiers. Les nouvelles possibilités font que cela n'a plus une importance cruciale.

Malheureusement, ces nouvelles fonctionnalités compliquent quelque peu la création de disquettes d'amorce les exploitant. Pour construire un système de fichiers compressé sur une disquette, celui-ci doit tout d'abord être créé sur un autre périphérique, puis compressé et transféré sur la disquette. Cela implique donc des étapes supplémentaires.

Schématiquement, il faut créer un système de fichiers compressé pour la racine, copier le noyau sur la disquette, dire à celui-ci où trouver le système de fichiers racine, puis copier celui-ci sur la disquette.

Voici un schéma simple en ASCII montrant la structure de la disquette :

```
|<--- zImage --->|<-- Systeme de fichiers racine compresse -->|
|________________|____________________________________________|
                    Disquette
```

Voici les étapes pour la création de la disquette amorce :

5.2 Création du système de fichiers racine

Le système de fichiers racine est créé en gros de la même manière qu'indiqué dans la section 4.1 de ce document. La différence principale vient du fait que vous ne pouvez plus créer le système de fichiers directement sur la disquette, mais que vous devez le créer sur un périphérique différent de plus grande taille que l'espace de la disquette qu'il va occuper.

5.2.1 Choix du périphérique

Pour construire un tel système de fichiers pour la racine, vous avez besoin d'un périphérique en plus de taille suffisante. Vous avez plusieurs choix :

- Si vous disposez d'une partition inutilisée sur un disque dur suffisamment grande (plusieurs megaoctets), c'est la solution la plus simple. Sinon, si vous avez suffisamment de mémoire vive physiquement présente, vous pouvez arrêter le swap et construire le système de fichiers dans la partition de swap.

  Cependant, la plupart des gens n'ont pas de partition en trop et ne peuvent pas non plus se permettre de retirer le swap, donc ...

- Utilisez un périphérique *loopback*. Un tel périphérique permet de faire passer un fichier présent sur un système de fichier existant pour un périphérique. Pour utiliser des périphériques *loopback*, vous devez utiliser des programmes `mount` et `umount` spécialement modifiés. Vous pouvez les trouver sur :

  ```
  ftp://ftp.win.tue.nl:/pub/linux/util/mount-2.5X.tar.gz
  ```

  où X représente la lettre de la dernière modification.

  Si vous n'avez pas de fichiers spéciaux correspondants aux périphériques *loopback* (`/dev/loop0`, `/dev/loop1`, etc) sur votre système, vous devrez d'abord les créer. Les commandes :

  ```
  mknod /dev/loop0 b 7 0
  mknod /dev/loop1 b 7 1
  mknod /dev/loop2 b 7 2
  ...
  ```

  vous permettront cela. Vous n'avez sans doute besoin que de `loop0`.

  Une fois ces binaires spéciaux de `mount` et `umount` installés, créez un fichier temporaire sur un disque dur de taille suffisante (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 PERIPHERIQUE ci-dessous. Quand vous lancez une commande `mount`, vous devez y ajouter l'option "`-o loop`" pour lui signaler qu'il doit utiliser un périphérique *loopback*. Par exemple :

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

  montera `/tmp/fsfile` (par périphérique *loopback*) sur le point de montage `/mnt`. Un "`df`" vous le confirmera.

- La dernière possibilité est d'utiliser le disque mémoire (PERIPHERIQUE = `/dev/ram0` ou `/dev/ramdisk`). Dans ce cas, la mémoire est utilisée pour simuler un disque. Le disque mémoire doit être assez grand pour contenir un système de fichiers de la taille nécessaire. Cherchez dans votre fichier de configuration pour LILO (`/etc/lilo.conf`) une ligne de la forme :

  ```
  RAMDISK_SIZE = nnn
  ```

qui détermine la quantité de mémoire vive à allouer. La valeur par défaut est 4096 Ko.

Après avoir choisi une de ces possibilités, préparez le périphérique avec :

```
dd if=/dev/zero of=PERIPHERIQUE bs=1k count=3000
```

Cette commande remplit de zéros le périphérique. Cette étape est importante, car le système de fichiers sur le périphérique étant compressé ultérieurement, toutes les parties inutilisées doivent être mises à zéro pour obtenir la meilleure compression possible.

Ensuite, créez le système de fichiers avec :

```
mke2fs -m 0 PERIPHERIQUE
```

(Si vous utilisez un périphérique *loopback*, le fichier que vous utilisez doit être fourni à la place de PERIPHERIQUE. `Mke2fs` vous demandera alors si vous voulez vraiment faire cela ; répondez oui.)

Puis montez le périphérique :

```
mount -t ext2 PERIPHERIQUE /mnt
```

Comme précédemment, copiez les fichiers dans `/mnt`, comme expliqué à la section 4.1.

5.2.2 Compression du système de fichiers

Une fois copiés les fichiers sur le système de fichiers racine, vous devez copiez celui-ci, puis le compresser. Tout d'abord, démontez le :

```
umount /mnt
```

(Techniquement, vous pouvez copiez le système de fichiers sans le démonter auparavant, mais cela peut être dangereux, et c'est de toute manière une mauvaise habitude.)

Puis, copiez les données depuis le périphérique dans un fichier. Appelez le fichier `rootfs` :

```
dd if=PERIPHERIQUE of=rootfs bs=1k
```

Puis, compressez le. Utilisez l'option "`-9`" de `gzip` pour obtenir une compression maximale :

```
gzip -9 rootfs
```

Cela peut prendre plusieurs minutes. Une fois fini, vous aurez un fichier `rootfs.gz` qui contient votre système de fichiers racine compressé.

Si vous manquez d'espace disque, vous pouvez combiner `dd` et `gzip` :

```
dd if=PERIPHERIQUE bs=1k | gzip -9 > rootfs.gz
```

5.3 Calcul de la taille

A présent, vérifiez la taille de l'ensemble afin de vérifier que le noyau ainsi que le système de fichiers racine tiendront sur la disquette. Un "`ls -l`" vous montrera combien d'octets occupent chacun ; divisez par 1024 pour obtenir le nombre de blocs dont chacun aura besoin. Pour les fractions de bloc, arrondissez au nombre supérieur.

Par exemple, si le noyau occupe 453281 octets, il nécessitera

```
ceil(453281 / 1024) = 443
```

blocs et occupera donc les blocs 0 à 442 sur la disquette. Le système de fichiers racine commencera au bloc 443. Notez ce numéro de bloc pour les commandes à suivre ; appelez le DEBUTRACINE.

Vous devez indiquer au noyau où il peut trouver le système de fichiers racine sur la disquette. A l'intérieur de l'image du noyau se trouve un mot pour le disque mémoire qui indique où se trouve la racine, ainsi que d'autres options. Le mot est défini dans `/usr/src/linux/arch/i386/kernel/setup.c` et est interprété comme suit :

```
bits  0-10~:        Decalage pour le debut du disque memoire, en
                    blocs de 1024 octets (c'est DEBUTRACINE,
                    calcule ci-dessus)
bits 11-13~:        inutilise
bit     14~:        drapeau indiquant que le disque memoire doit
                    etre charge
bit     15~:        drapeau indiquant qu'il faut demander
                    l'insertion de la disquette
```

(Si le bit 15 est positionné à 1, on vous demandera au démarrage de placer une nouvelle disquette dans le lecteur. C'est nécessaire pour un kit de démarrage en deux disquettes, décrit ci-dessous dans la section "Fabriquer un kit sur deux disques". Pour l'instant, considérons le à 0).

Si le système de fichiers racine doit commencer au bloc 443, le mot du disque mémoire est :

```
   1BB (hexa) 443 (decimal)                           (bits 0-10)
+ 4000 (hexa) Drapeau de chargement du disque memoire (bit    14)
  -----------
= 41BB (hexa)
=16827 (decimal)
```

Ce mot doit être positionné dans l'image du noyau en utilisant la commande "`rdev -r`" dans la section suivante.

5.4 Copie des fichiers sur la disquette

Vous êtes maintenant prêt à créer la disquette d'amorce. Tout d'abord, copiez le noyau :

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

Ensuite, indiquez à celui-ci où trouver son système de fichiers racine sur la disquette :

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

Vous devez maintenant positionner le mot du disque mémoire dans l'image du noyau située sur la disquette. On donne sa valeur au mot en utilisant la commande “`rdev -r`”. En prenant la valeur calculée ci-dessus dans la section intitulée “Calcul de la taille”, cela donne :

```
rdev -r /dev/fd0 16827
```

Enfin, placez le système de fichiers racine sur la disquette derrière le noyau. La commande `dd` possède une option `seek` vous permettant d'indiquer combien de blocs sauter avant d'écrire :

```
dd if=rootfs.gz of=/dev/fd0 bs=1k seek=443
```

(Le nombre 443 correspond à DEBUTRACINE provenant de la section “Calcul de la taille” précédente.)

Attendez que l'écriture soit finie sur la disquette, et vous êtes prêt.

5.5 Fabriquer un kit sur deux disques

Si vous avez besoin de plus de place, vous pouvez fabriquer un kit d'amorce sur deux disques. Dans ce cas, la première disquette contiendra uniquement le noyau, et la seconde le système de fichiers racine compressé. Dans cette configuration, votre système racine compressé peut faire jusqu'à 1440 Ko.

La création d'un ensemble de deux disquettes d'amorce est réalisée de manière sensiblement identique aux étapes précédemment décrites. Tout d'abord, vous devez positionner le drapeau de demande de disquette mémoire à 1 pour indiquer au noyau qu'il faut demander la seconde disquette et attendre son insertion. Le système racine commence à l'octet 0 de la second disquette.

Comme indiqué dans la précédente section “Calcul de la taille”, le drapeau indiquant qu'il faut demander l'insertion de la disquette (bit 15) doit être positionné à 1, et le décalage du disque mémoire sera zéro. Dans notre exemple, le calcul donnera cette fois :

```
  4000 (hexa) drapeau de chargement du disque memoire (bit 14)
+ 8000 (hexa) drapeau demandant une invite pour le
              chargement de la disquette racine       (bit 15)
------------
= C000 (hexa)
=49152 (decimal)
```

qui sera utilisé dans le calcul pour “`rdev -r`” comme précédemment.

Suivez les instructions de “Copie des fichiers sur la disquette” indiquées ci-dessus, mais après avoir lance la commande “`rdev -r`”, insérez une nouvelle disquette dans le lecteur et lancez la commande :

```
dd if=rootfs.gz of=/dev/fd0
```

L'option `seek` n'est pas nécessaire puisque le système de fichiers racine démarre au bloc zéro.

6 En cas de problèmes

En construisant des disques de secours, il est courant que les premiers essais ne démarrent pas. En général, pour construire un disque d'amorce, on assemble des composants de son système existant, et on tente d'obtenir que le système sur disquette démarre jusqu'à afficher des messages sur la console. Une fois qu'il vous parle, vous avez presque gagné la bataille, car vous pouvez voir de quoi il se plaint, et corriger les problèmes individuels jusqu'à ce que le système marche sans problèmes. Si le système s'arrête sans explications, il peut être difficile de trouver l'origine du problème. Pour obtenir un système qui démarre et arrive à parler, un certain nombre de composants doivent être présents et correctement configurés. La méthode recommandée pour trouver le problème si le système n'affiche rien est comme suit :

- Vérifiez que le disque racine contient bien les répertoires que vous croyez. Il est facile de se tromper de répertoire origine et de se retrouver avec quelque chose du genre `/disque_racine/bin` au lieu de `/bin` sur votre disquette racine.
- Vérifiez que `/lib/libc.so` et `/lib/libtermcap.so` sont présents, avec les mêmes liens que dans votre répertoire lib sur le disque dur.
- Vérifiez que tous les liens symboliques du répertoire `/dev` de votre système existent également dans votre système de fichiers racine, quant il s'agit de liens vers des périphériques que vous avez inclus dans votre disquette racine. En particulier, `/dev/console` est indispensable dans bien des cas.
- Vérifiez que vous avez inclus `/dev/tty1` sur votre disque racine.
- Vérifiez que vous avez inclus les périphériques `/dev/null`, `/dev/zero`, `/dev/mem`, `/dev/ram` et `/dev/kmem`.
- Vérifiez la configuration de votre noyau - le support de toutes les ressources nécessaires pour vous logguer doivent être inclus dans le noyau, et non pas sous forme de modules. Surtout, *le support pour disque mémoire (*`ramdisk`*) doit être inclus*.
- Vérifiez que les paramètres de périphérique racine et de disque mémoire du noyau sont corrects. Reportez-vous à la section 5 pour les détails.

Une fois ces divers points généraux couverts, voici quelques points plus particuliers à surveiller :

1. Vérifiez qu'init est présent en tant que `/sbin/init` ou `/bin/init`, et qu'il est executable.
2. Lancez `ldd init` pour vérifier les bibliothèques d'`init`. En général il n'y a que `libc.so`, mais il vaut mieux vérifier. Vérifiez que vous incluez ces bibliothèques.
3. Lancez `file` sur la(les) bibliothèque(s) indiquées par ldd pour connaître leur type. Vérifiez que vous avez le bon chargeur présent sur le disque racine. Le chargeur est soit `ld.so` (pour les bibliothèques *a.out*), soit `ld-linux.so` (pour les bibliothèques *ELF*).
4. Vérifiez `/etc/inittab` sur le système de fichiers de votre disque amorce pour les appels a *getty*.

   La notation `*getty*` sera utilisée pour indiquer des programmes du genre de `getty`, comme `getty`, `agetty`, `mgetty` ou `getty_ps`.

Vérifiez bien ceux-ci avec ceux de l'`inittab` de votre disque dur. Vérifiez les pages de manuel du programme pour vous assurez que cela est utile. `Inittab` est peut-être la partie la plus difficile car sa syntaxe et son contenu dépendent du programme `init` utilisé et de la nature du système. La seule façon de résoudre le problème est de lire les pages de manuel d'`init` et `inittab` et de comprendre ce que fait exactement votre système existant lorsqu'il démarre. Vérifiez que `/etc/inittab` a une entrée pour l'initialisation du système. Elle devrait contenir une commande de la forme `/etc/rc.x`, pour lancer l'un des scripts `/etc/rc`. Ce script doit exister.

5. Pour `init`, lancez `ldd` sur `getty` (ou `agetty`) pour déterminer ses besoins, et vérifiez que les bibliothèques et chargeurs nécessaires ont été inclus dans votre système de fichiers racine.

6. Si vous avez un fichier `/etc/ld.so.cache` sur votre disque de secours, regénérez-le.

Si `init` démarre, mais que vous obtenez un message du type :

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

cela provient d'`init`, qui indique généralement par là que votre `*getty*` ou `login` se termine aussitôt démarré. Vérifiez les exécutables de `*getty*` et `login`, et les bibliothèques dont ils dépendent. Vérifiez que les invocations dans `/etc/inittab` sont correctes. Si vous obtenez d'étranges messages de `*getty*`, cela peut vouloir dire que son appel dans `/etc/inittab` est erroné. Les options des programmes `*getty*` sont variées ; même entre différentes versions de `agetty`, il semblerait qu'on trouve des arguments incompatibles. Si vous utilisez des arguments ou un programme différent de ce que vous utilisez dans le `/etc/inittab` de votre disque dur, vérifiez plutôt deux fois qu'une.

Si vous essayez de lancer un programme quelconque, tel que `df`, qui se trouve sur votre disque de secours mais que vous obtenez un message du genre : `df: not found`, vérifiez deux choses :

1. Vérifiez que le répertoire contenant le programme est dans votre variable `PATH`

2. Vérifiez que vous avez les bibliothèques et chargeurs dont le programme a besoin. Tapez `ldd fichier` pour voir la liste des bibliothèques nécessaires, et vérifiez que celles-ci existent. Voir la section précédente sur `/lib`.

7 Foire Aux Questions (FAQ)

7.1 Q. Je démarre à partir de mes disques amorce et racine et rien ne se passe. Que dois-je faire ?

Cette réponse a été transférée dans la Section 6 ci-dessus.

7.2 Q. Comment puis-je faire un disque d'amorce avec un gestionnaire pour XXX ?

La manière la plus facile est d'obtenir le noyau d'une distribution Slackware depuis le site miroir Slackware le plus proche. Les noyaux Slackware sont des noyaux génériques qui tentent d'inclure autant de gestionnaires pour des périphériques différents que possible donc si vous avez un controlleur SCSI ou IDE, il y a de grandes chances qu'un gestionnaire approprié soit inclus dans le noyau Slackware.

Allez dans le répertoire `a1` et sélectionnez un noyau soit SCSI, soit IDE, en fonction du type de controlleur que vous avez. Vérifiez dans le fichier `xxxxkern.cfg` correspondant au noyau que vous avez sélectionné la liste des gestionnaires inclus dans ce noyau. Si le périphérique dont vous avez besoin se trouve dans la liste, alors le noyau correspondant devrait pouvoir démarrer votre ordinateur. Récupérez le fichier `xxxxkern.tgz` et copiez-le sur votre disquette d'amorce comme indiqué précédemment dans la section sur la fabrication des disques d'amorces.

Vous devez alors indiquer le périphérique de démarrage au noyau, en utilisant la commande `rdev` :

```
rdev zImage
```

`Rdev` va alors indiquer le périphérique racine actuel configuré dans le noyau. S'il ne s'agit pas de celui que vous voulez, utilisez alors `rdev` pour le modifier. Par exemple, le noyau que j'ai utilisé était réglé pour `/dev/sda2`, mais ma partition SCSI racine est `/dev/sda8`. Pour utiliser une disquette racine, vous devriez lancer la commande :

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

Si vous voulez aussi savoir comment utiliser une disquette racine Slackware, c'est un sujet hors du cadre de ce HOWTO, et je suggère donc que vous lisiez le Guide d'Installation de Linux ou que vous récupériez la distribution Slackware. Cf. la section de ce HOWTO intitulée “Références”.

7.3 Q. Comment puis-je mettre à jour ma disquette d'amorce avec un nouveau noyau ?

Il suffit de copier le noyau sur la disquette amorce en utilisant la commande `dd` dans le cas d'une disquette amorce sans système de fichiers, ou alors d'utiliser la commande `cp` dans le cas d'un ensemble de disques amorce et racine. Reportez-vous à la section de ce HOWTO intitulée “Amorce” pour plus de détails sur la création d'un disque d'amorce. La description s'applique aussi à la mise à jour d'un noyau sur un disque amorce.

7.4 Q. Comment puis-je enlever LILO pour utiliser à nouveau DOS pour démarrer l'ordinateur ?

Ce n'est pas vraiment un sujet sur les disques d'amorce, mais la question est souvent posée, et donc : la réponse est d'utiliser la commande DOS :

```
FDISK /MBR
```

MBR signifie *Master Boot Record* (enregistrement original d'amorce), et cela remplace le secteur d'amorce par un original du DOS, sans affecter la table de partitions. Quelques puristes ne semblent pas d'accord, mais même l'auteur de LILO, Werner Almesberger, le suggère. C'est simple, et cela fonctionne.

Vous pouvez aussi utiliser la commande dd pour copier la sauvegarde effectuée par LILO vers le secteur d'amorce : voir la documentation de LILO si vous souhaitez le faire.

7.5 Q. Comment puis-je démarrer si j'ai perdu mon noyau ET mon disque amorce ?

Si vous n'avez pas de disque amorce sous la main, la solution la plus simple est sans doute de récupérer un noyau Slackware correspondant à votre type de controlleur disque (IDE ou SCSI), comme décrit ci-dessus dans la réponse à la question "Comment puis-je faire un disque d'amorce avec un gestionnaire pour XXX ?". Vous pouvez alors démarrer votre ordinateur en utilisant ce noyau, puis réparer les dégâts.

Le noyau récupéré peut ne pas être configuré pour s'amorcer sur le type de disque ou sur la partition que vous souhaitez. Par exemple, le noyau générique SCSI de Slackware est configuré pour démarrer avec comme périphérique racine /dev/sda2, alors que ma partition racine Linux se trouve être /dev/sda8. Dans ce cas, il faut modifier la configuration du périphérique racine dans le noyau.

Il reste possible de changer la racine ou les paramètres du disque-mémoire dans le noyau même si tout ce dont vous disposez est un noyau, et un autre système d'exploitation tel que DOS.

Rdev modifie la configuration du noyau en changeant les valeurs situées à certaines adresses dans le fichier du noyau, ce que vous pouvez faire si vous avez un éditeur hexadécimal disponible dans l'un des systèmes qui marche encore - par exemple, le Disk Editor des Norton Utilities sous DOS. Vous devez alors vérifier et éventuellement modifier les valeurs contenues dans le noyau aux adresses suivantes :

```
0x01F8  Octet de poids faible de la taille du disque-memoire
0x01F9  Octet de poids fort de la taille du disque-memoire
0x01FC  Numero de peripherique mineur de la racine_- voir ci-dessous
0X01FD  Numero de peripherique majeur de la racine_- voir ci-dessous
```

La taille du disque mémoire est le nombre de blocs à créer pour le disque. Si vous désirez démarrer depuis une disquette racine, il faut mettre cette valeur a 1440 en base décimale, soit 0x05A0, et donc mettre la valeur 0xA0 au décalage 0x01F8 et la valeur 0x05 au décalage 0x01F9. La place sera allouée pour une disquette de 1.4 Mo.

Notons que la signification du mot donnant la taille du disque mémoire a changé dans la version 1.3.48 du noyau. Cette signification est décrite dans la section 5.

Les numéros de périphérique majeurs et mineurs doivent être positionnés en fonction du périphérique que vous souhaitez utiliser comme système de fichiers racine. Voici un certain nombre de valeurs utiles :

```
Peripherique   majeur mineur
/dev/fd0            2      0   premier lecteur de disquette
/dev/hda1           3      1   partition 1 sur le premier disque IDE
/dev/sda1           8      1   partition 1 sur le premier disque SCSI
/dev/sda8           8      8   partition 8 sur le premier disque SCSI
```

Une fois ces valeurs entrées, vous pouvez écrire le fichier sur une disquette en utilisant soit le Disk Editor des Norton Utilities, soit un programme appelé rawrite.exe. Ce programme est inclus dans de nombreuses distributions, comme la SLS et la Slackware. Il s'agit d'un programme DOS qui écrit un fichier vers les "couches basses" du disque, en partant du secteur d'amorce, au lieu de l'écrire sur le système de fichiers. Si vous utilisez les Norton Utilities, vous devez alors faire écrire le fichier sur un disque physique en commençant au début du disque.

7.6 Q. Comment puis-je faire des copies supplémentaires des disquettes amorce et racine ?

Il n'est jamais souhaitable de n'avoir qu'un seul jeu de disques de secours - 2 ou 3 devraient être conservés au cas ou l'un est illisible.

Le plus simple pour réaliser des copies de disquettes, quelles qu'elles soient, y compris des disquettes amorçables et d'utilitaires, est d'utiliser la commande `dd` pour copier le contenu de la disquette originale dans un fichier sur votre disque dur, puis d'utiliser la même commande pour recopier ce fichier sur une nouvelle disquette. Notez que vous n'avez pas besoin, et ne devriez pas en avoir besoin, de monter les disquettes, car `dd` utilise l'interface d'accès au périphérique physique.

Pour copier l'original, tapez la commande :

```
dd if=nom-du-peripherique of=nom-du-fichier
ou      nom-du-peripherique est le nom du peripherique du
        lecteur de disquette
et      nom-du-fichier est le nom du fichier dans lequel vous
        voulez ecrire
```

Par exemple, pour copier depuis `/dev/fd0` vers un fichier temporaire appelé `/tmp/disquette.copie`, je taperais la commande :

```
dd if=/dev/fd0 of=/tmp/disquette.copie
```

Ne pas donner le paramètre "`count`", comme ici, signifie que l'ensemble de la disquette, soit 2880 blocs pour une disquette haute densité, seront copiés.

Pour recopier le fichier résultant sur une nouvelle disquette, mettez la disquette dans le lecteur et tapez la commande inverse :

```
dd if=fnom-du-fichier of=nom-du-peripherique
```

Notons que l'on fait ici l'hypothèse que vous n'avez qu'un seul lecteur de disquette. Si vous en avez deux du même type, vous pouvez copier des disquettes avec une commande du type :

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

7.7 Q. Comment puis-je démarrer sans avoir à taper "ahaxxxx=nn,nn,nn" à chaque fois ?

Quand un périphérique de disque n'est pas détecté automatiquement au démarrage, il faut fournir au noyau une chaîne de paramètres pour le périphérique, telle que :

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

Cette chaîne peut être fournie de diverses manières grâce à LILO :

- En la tapant sur la ligne de commande à chaque fois que l'on démarre le système grâce à LILO. Cela devient vite lassant, cela dit.
- En utilisant le mot clé "`lock`" de LILO pour s'assurer qu'il conserve la ligne de commande en question comme ligne de commande par défaut, utilisant ainsi les mêmes options à chaque démarrage.
- En utilisant l'instruction `APPEND` dans le fichier de configuration de LILO. Dans ce cas la chaîne de paramètres doit être entourée de guillemets.

Ainsi, un exemple de ligne de commande utilisant la chaîne de paramètres ci-dessus pourrait être :

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

La chaîne de paramètres pour le périphérique est ainsi passée au noyau, tout en lui indiquant d'utiliser `/dev/sda1` comme périphérique racine et de sauver la ligne de commande complète afin de la réutiliser pour tous les futurs démarrages.

Un exemple d'utilisation de l'instruction `APPEND` serait :

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

A noter que la chaîne NE DOIT PAS être entourée de guillemets dans le cas de la ligne de commande, mais DOIT l'être lors de l'utilisation d'`APPEND`.

Notons aussi que pour que la chaîne de paramètres soit prise en compte par le noyau, celui-ci doit contenir le gestionnaire pour ce type de disque. Si ce n'est pas le cas, il n'y a alors personne à l'écoute des arguments fournis, et vous devrez reconstruire le noyau pour inclure le gestionnaire requis. Pour plus de détails sur la construction d'un noyau, allez dans le répertoire `/usr/src/linux` et lisez le fichier README, puis la FAQ Linux et le HOWTO Installation. Ou encore vous pouvez obtenir un noyau générique pour le type de disque utilisé et l'installer à la place.

Il est incessamment recommandé aux lecteurs de lire la documentation de LILO avant de faire des essais d'installation. L'utilisation de l'instruction `BOOT` peut en particulier endommager des partitions si elle tombe mal à point.

7.8 Q. Comment puis-je créer un système de fichiers en mémoire de grande taille ?

Pour les noyaux supérieurs à la version 1.3.48, il vaut mieux créer un système de fichiers compressé comme indiqué dans la section 5. Si votre noyau est antérieur à cette version vous pouvez soit le mettre à jour, soit vous referer à la version 2.0 ou antérieure de ce HOWTO.

7.9 Q. Au démarrage, j'obtiens l'erreur `a: cannot execute B`. Pourquoi ?

Il arrive souvent que les noms de programmes soient écrits en dur dans divers utilitaires. Cela n'arrive pas partout, mais cela peut expliquer pourquoi un executable ne peut apparemment pas être trouvé sur votre système de fichiers même lorsque vous le voyez vous-même. Vous pouvez savoir si un programme donné contient le nom d'un autre programme écrit directement grâce à la commande "`strings`", en redirigeant sa sortie vers le programme `grep`.

Quelques exemples de codage en dur :

- `Shutdown` dans certaines versions contient `/etc/reboot` en dur, nécessitant que vous placiez `reboot` dans le répertoire `/etc`.
- `Init` a posé des problèmes à au moins une personne pour qui le noyau ne pouvait pas trouver `init`.

Pour résoudre ces problèmes, déplacez les programmes vers le répertoire attendu, ou changez les fichiers de configuration (par exemple `inittab`) pour qu'ils pointent vers les répertoires corrects. Dans le doute, mettez les programmes dans les mêmes répertoires que sur votre disque dur, et utilisez le même fichier `inittab` et les mêmes fichiers de `/etc/rc.d` que sur votre disque.

7.10 Q. Mon noyau a le support pour disque mémoire, mais initialise des disques mémoires de 0 ko.

Lorsque cela arrive, un message du type :

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

apparaît lors du démarrage du noyau. La taille doit être soit la taille par défaut, soit 4096 ko, soit la taille spécifiée dans les paramètres du noyau `ramdisk_size` ou `ramdisk`. Si vous obtenez une taille de 0 ko, c'est probablement parce que celle-ci a été fixée à 0 par les paramètres du noyau au démarrage. Cela peut être du à une ligne oubliée dans le fichier de configuration de LILO, du type

```
ramdisk 0
```

Cette ligne était incluse dans les fichiers exemples de configuration de LILO fournis avec d'anciennes distributions, afin d'écraser tout paramètre modifié du noyau. Depuis la version 1.3.48 c'est inutile, puisque le paramètre `ramdisk_size` du noyau indique maintenant la taille maximum du disque mémoire, et non pas la taille allouée au démarrage, aucun disque mémoire n'étant alloué au démarrage.

La solution est de retirer le paramètre `ramdisk` de LILO.

A noter que si vous tentez d'utiliser un disque mémoire dont la taille est de 0 ko, vous obtenez un comportement imprévisible pouvant résulter en panique du noyau.

A Ressources et liens

Dans cette section, *vvv* sera utilisé dans les noms de paquetages à la place du numéro de version, afin d'éviter de référencer ici des versions spécifiques. Lorsque vous récupérez un paquetage, récupérez toujours la dernière version, sauf si vous avez de bonnes raisons de ne pas le faire.

A.1 Disques amorces de distributions

Ce sont les principales sources de disques amorces pour les distributions.

Utilisez de préférence un des sites miroirs afin de réduire la charge de ces machines.

- *Disques amorces Slackware* (`http://sunsite.unc.edu/pub/Linux/distributions/slackware/bootdsks.144/`) et *Sites miroirs Slackware* (`http://sunsite.unc.edu/pub/Linux/distributions/slackware/MIRRORS.TXT`)
- *Disques amorces Red Hat* (`http://sunsite.unc.edu/pub/Linux/distributions/redhat/current/i386/images/`) et *Sites miroirs Red Hat* (`http://www.redhat.com/ftp.html`)
- *Disques amorces Debian* (`ftp://ftp.debian.org/pub/debian/stable/disks-i386`) et *Sites miroirs Debian* (`ftp://ftp.debian.org/debian/README.mirrors`)

A.2 LILO - Chargeur Linux

Ecrit par Werner Almesberger. Excellent chargeur d'amorce, dont la documentation contient des informations sur le contenu du secteur d'amorce et les étapes initiales du processus d'amorce.

FTP depuis :`tsx-11.mit.edu: /pub/linux/packages/lilo/lilo.vvv.tar.gz` et aussi disponible sur sunsite et ses sites miroirs.

A.3 FAQ Linux et HOWTO

Ils sont disponibles depuis de nombreuses sources. Lisez les forums de discussion Usenet `news.answers` et `comp.os.linux.announce`.

FTP depuis : `sunsite.unc.edu:/pub/Linux/docs`

- La FAQ est dans `/pub/linux/docs/faqs/linux-faq`
- Les HOWTO sont dans `/pub/Linux/docs/HOWTO`

Pour le WWW, démarrez sur la page racine de la documentation Linux :

`http://sunsite.unc.edu/mdw/linux.html`

Si vous êtes vraiment désespéré, envoyez un courrier électronique à :

`mail-server@rtfm.mit.edu`

avec le mot “`help`” dans le corps du message, puis suivez les instructions reçues par retour du courrier.

Note : si vous n’avez pas lu la FAQ Linux et les documents annexes tels que le HOWTO sur l’installation de Linux et le Guide d’Installation de Linux, vous ne devriez pas essayer de construire des disquettes d’amorce.

A.4 Utilisation de disque mémoire

Une excellente description de la manière de fonctionner du nouveau code de disque mémoire se trouve dans la documentation fournie avec le noyau Linux. Voir `/usr/src/linux/Documentation/ramdisk.txt`. Il s’agit d’un texte écrit par Paul Gortmaker, contenant une section sur la création de disque mémoire compressé, similaire à la section 5 de ce HOWTO.

A.5 Paquetages de secours

A.5.1 Bootkit

Ecrit par Scott Burkett. Bootkit fournit une base souple, gérée par menus, permettant la création de disques de secours et la gestion de leur contenu. Il utilise le paquetage Dialog pour fournir de sympathiques menus, et une arborescence de répertoire simple pour contenir les définitions du contenu des disques de secours. Le paquetage contient également des exemples pour les principaux fichiers utilisés. Il ne vise qu’à fournir la base cependant ; c’est à l’utilisateur de déterminer ce qu’il veut mettre sur les disques et comment mettre en place les fichiers de configuration en conséquence. Pour les utilisateurs que cela ne dérange pas, c’est un bon choix.

FTP depuis : `sunsite.unc.edu:/pub/Linux/system/Recovery/Bootkit-vvv.tar.gz`

A.5.2 CatRescue

Ecrit par Oleg Kibirev. Ce paquetage se concentre surtout sur l’économie de place sur les disquettes de secours par l’utilisation extensive de la compression, et en implémentant des exécutables avec des scripts shell. La documentation contient quelques astuces sur la conduite à tenir dans divers cas graves.

FTP depuis : `gd.cs.csufresno.edu/pub/sun4bin/src/CatRescue100.tgz`

A.5.3 Rescue Shell Scripts

Ecrit par Thomas Heiling. Il contient des scripts shell permettant de créer des disquettes d’amorce, ainsi que des ensembles disquette d’amorce et disquette racine. Il nécessite l’installation de certaines versions spécifiques d'autres logiciels tels que LILO, et peut vous demander quelques efforts pour l'adapter à votre système, mais il peut servir de point de départ si vous cherchez des scripts shell plus complets que ceux fournis dans ce document.

FTP depuis : `sunsite.unc.edu:/pub/Linux/system/Recovery/rescue.tgz`

A.5.4 SAR - Search And Rescue

Ecrit par Karel Kubat. SAR crée une disquette de secours, en utilisant divers moyens pour minimiser l'espace disque requis sur la disquette. Le manuel comprend une description du processus d'amorce et de login sous Linux.

FTP depuis : `ftp.icce.rug.nl:/pub/unix/SAR-vvv.tar.gz`

Le manuel est disponible par WWW depuis :

`http://www.icce.rug.nl/karel/programs/SAR.html`

A.5.5 YARD

Ecrit par Tom Fawcett. Yard produit des disquettes de secours personnalisées en utilisant l'option des noyaux récents (postérieurs à 1.3.48) permettant d'utiliser des disques mémoire compressés. Yard a été conçu pour automatiser la plupart des étapes de la section 5 ci-dessus. De plus, Yard vérifie vos choix de fichiers (chargeurs, bibliothèques, `/etc/fstab`, `rc`, `/etc/passwd`) pour s'assurer que vous avez inclus tout ce qui est nécessaire pour créer un disque de secours amorçable. Yard nécessite Perl 5 et un noyau de version postérieure à 1.3.48.

La page racine de Yard se trouve à `(http://www.cs.umass.edu/ fawcett/yard.html)`, qui devrait toujours pointer vers la dernière version, ainsi que des notes sur tout bug récent. Yard peut aussi être récupéré depuis `(http://sunsite.unc.edu/pub/Linux/system/Recovery/)`.

B Exemples

B.1 Contenu des répertoires du disque

Ceci est le contenu des répertoires de mes disquettes racine et utilitaire. Ces listes sont uniquement fournies en tant qu'exemple des fichiers inclus pour créer un système fonctionnel. J'ai ajouté quelques notes là où cela semblait utile.

B.1.1 Contenu des répertoires du disque racine (ls -lR)

```
total 18
drwxr-xr-x   2 root     root         1024 Jul 29 21:16 bin/
drwxr-xr-x   2 root     root         9216 Jul 28 16:21 dev/
drwxr-xr-x   3 root     root         1024 Jul 29 20:25 etc/
drwxr-xr-x   2 root     root         1024 Jul 28 19:53 lib/
```

```
drwxr-xr-x   2 root     root         1024 Jul 24 22:47 mnt/
drwxr-xr-x   2 root     root         1024 Jul 24 22:47 proc/
drwxr-xr-x   2 root     root         1024 Jul 28 19:07 sbin/
drwxr-xr-x   2 root     root         1024 Jul 29 20:57 tmp/
drwxr-xr-x   4 root     root         1024 Jul 29 21:35 usr/
drwxr-xr-x   3 root     root         1024 Jul 28 19:52 var/

bin:
total 713
-rwxr-xr-x   1 root     bin          7737 Jul 24 22:16 cat*
-rwxr-xr-x   1 root     bin          9232 Jul 24 22:48 chmod*
-rwxr-xr-x   1 root     bin          8156 Jul 24 22:48 chown*
-rwxr-xr-x   1 root     bin         19652 Jul 24 22:48 cp*
-rwxr-xr-x   1 root     root         8313 Jul 29 21:16 cut*
-rwxr-xr-x   1 root     bin         12136 Jul 24 22:48 dd*
-rwxr-xr-x   1 root     bin          9308 Jul 24 22:48 df*
-rwxr-xr-x   1 root     root         9036 Jul 29 20:24 dircolors*
-rwxr-xr-x   1 root     bin          9064 Jul 24 22:48 du*
-rwxr-x---   1 root     bin         69252 Jul 24 22:51 e2fsck*
-rwxr-xr-x   1 root     bin          5361 Jul 24 22:48 echo*
-rwxr-xr-x   1 root     bin          5696 Jul 24 22:16 hostname*
-rwxr-xr-x   1 root     bin          6596 Jul 24 22:49 kill*
-rwxr-xr-x   1 root     bin         10644 Jul 24 22:17 ln*
-rwxr-xr-x   1 root     bin         13508 Jul 24 22:17 login*
-rwxr-xr-x   1 root     bin         26976 Jul 24 22:17 ls*
-rwxr-xr-x   1 root     bin          7416 Jul 24 22:49 mkdir*
-rwxr-x---   1 root     bin         34596 Jul 24 22:51 mke2fs*
-rwxr-xr-x   1 root     bin          6712 Jul 24 22:49 mknod*
-rwxr-xr-x   1 root     bin         20304 Jul 24 22:17 more*
-rwxr-xr-x   1 root     bin         24704 Jul 24 22:17 mount*
-rwxr-xr-x   1 root     bin         12464 Jul 24 22:17 mv*
-rwxr-xr-x   1 root     bin         20829 Jul 24 22:50 ps*
-rwxr-xr-x   1 root     bin          9424 Jul 24 22:50 rm*
-rwxr-xr-x   1 root     bin          4344 Jul 24 22:50 rmdir*
-rwxr-xr-x   1 root     root       299649 Jul 27 14:12 sh*
-rwxr-xr-x   1 root     bin          9853 Jul 24 22:17 su*
-rwxr-xr-x   1 root     bin           380 Jul 27 14:12 sync*
-rwxr-xr-x   1 root     bin         13620 Jul 24 22:17 umount*
-rwxr-xr-x   1 root     root         5013 Jul 29 20:03 uname*

dev:
total 0
lrwxrwxrwx   1 root     root           10 Jul 24 22:34 cdrom -> /dev/sbpcd
crw--w--w-   1 root     tty         4,   0 Jul 24 21:49 console
brw-rw----   1 root     floppy      2,   0 Apr 28  1995 fd0
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 ftape -> rft0
crw-rw-rw-   1 root     sys        10,   2 Jul 18  1994 inportbm
crw-rw----   1 root     kmem        1,   2 Jul 28 16:21 kmem
```

```
crw-rw----   1 root     kmem       1,   1 Jul 18  1994 mem
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 modem -> cua0
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 mouse -> cua1
crw-rw-rw-   1 root     sys        1,   3 Jul 18  1994 null
brw-rw----   1 root     disk       1,   1 Jul 18  1994 ram
crw-rw----   1 root     disk      27,   0 Jul 18  1994 rft0
brw-rw----   1 root     disk      25,   0 Jul 19  1994 sbpcd
** Je n'ai inclus les peripheriques que pour les partitions SCSI que
*** j'utilise. Si vous utilisez de l'IDE, utilisez /dev/hdxx a la place.
brw-rw----   1 root     disk       8,   0 Apr 29  1995 sda
brw-rw----   1 root     disk       8,   6 Apr 29  1995 sda6
brw-rw----   1 root     disk       8,   7 Apr 29  1995 sda7
brw-rw----   1 root     disk       8,   8 Apr 29  1995 sda8
lrwxrwxrwx   1 root     root            7 Jul 28 12:56 systty -> console
*** ce lien de systty vers console est necessaire
crw-rw-rw-   1 root     tty        5,   0 Jul 18  1994 tty
crw--w--w-   1 root     tty        4,   0 Jul 18  1994 tty0
crw--w----   1 root     tty        4,   1 Jul 24 22:33 tty1
crw--w----   1 root     tty        4,   2 Jul 24 22:34 tty2
crw--w--w-   1 root     root       4,   3 Jul 24 21:49 tty3
crw--w--w-   1 root     root       4,   4 Jul 24 21:49 tty4
crw--w--w-   1 root     root       4,   5 Jul 24 21:49 tty5
crw--w--w-   1 root     root       4,   6 Jul 24 21:49 tty6
crw-rw-rw-   1 root     tty        4,   7 Jul 18  1994 tty7
crw-rw-rw-   1 root     tty        4,   8 Jul 18  1994 tty8
crw-rw-rw-   1 root     tty        4,   9 Jul 19  1994 tty9
crw-rw-rw-   1 root     sys        1,   5 Jul 18  1994 zero

etc:
total 20
-rw-r--r--   1 root     root         2167 Jul 29 20:25 DIR_COLORS
-rw-r--r--   1 root     root           20 Jul 28 12:37 HOSTNAME
-rw-r--r--   1 root     root          109 Jul 24 22:57 fstab
-rw-r--r--   1 root     root          271 Jul 24 22:21 group
-rw-r--r--   1 root     root         2353 Jul 24 22:27 inittab
-rw-r--r--   1 root     root            0 Jul 29 21:02 issue
-rw-r--r--   1 root     root         2881 Jul 28 19:38 ld.so.cache
*** De nombreuses choses ne marchent pas au demarrage si ld.so.cache
*** est absent, mais verifiez que ldconfig est present et lance depuis
*** rc.x pour le mettre a jour.
-rw-r--r--   1 root     root           12 Jul 24 22:22 motd
-rw-r--r--   1 root     root          606 Jul 28 19:25 passwd
-rw-r--r--   1 root     root         1065 Jul 24 22:21 profile
drwxr-xr-x   2 root     root         1024 Jul 29 21:01 rc.d/
-rw-r--r--   1 root     root           18 Jul 24 22:21 shells
-rw-r--r--   1 root     root          774 Jul 28 13:43 termcap
-rw-r--r--   1 root     root          126 Jul 28 13:44 ttys
-rw-r--r--   1 root     root            0 Jul 24 22:47 utmp
```

```
etc/rc.d:
total 5
*** Je ne me suis pas encombre des scripts pour l'extinction_- tout
*** tourne dans un disque memoire, qui ne presente que peu d'interet
*** pour une procedure d'extinction (shutdown)
-rwxr-xr-x   1 root     root         1158 Jul 24 22:23 rc.K*
-rwxr-xr-x   1 root     root         1151 Jul 28 19:08 rc.M*
-rwxr-xr-x   1 root     root          507 Jul 29 20:25 rc.S*

lib:
total 588
*** J'ai un systeme ELF, et donc j'inclus mon chargeur ELF
*** ld-linux.so. Si vous etes toujours en a.out, vous avez besoin de
*** ld.so. Utilisez la commande file pour savoir de quelles
*** bibliotheques vous avez besoin.
lrwxrwxrwx   1 root     root           17 Jul 24 23:36 ld-linux.so.1 -> ld-linux.so.1.7.3*
-rwxr-xr-x   1 root     root        20722 Aug 15  1995 ld-linux.so.1.7.3*
lrwxrwxrwx   1 root     root           13 Jul 24 23:36 libc.so.5 -> libc.so.5.0.9*
-rwxr-xr-x   1 root     root       562683 May 19  1995 libc.so.5.0.9*
*** Il faut inclure libtermcap
lrwxrwxrwx   1 root     root           19 Jul 28 19:53 libtermcap.so.2 -> libtermcap.so.2.0.0*
-rwxr-xr-x   1 root     root        11360 May 19  1995 libtermcap.so.2.0.0*

mnt:
total 0

proc:
total 0

sbin:
total 191
*** J'utilise Slackware, qui utilise agetty. De nombreux systemes
*** utilisent getty. Verifiez /etc/inittab pour savoir lequel il
*** utilise. Notez bien que vous avez besoin de (a)getty et de login
*** pour esperer pouvoir faire quelque chose.
-rwxr-xr-x   1 root     bin         11309 Jul 24 22:54 agetty*
-rwxr-xr-x   1 root     bin          5204 Jul 24 22:19 halt*
*** Necessaire pour demarrer
-rwxr-xr-x   1 root     bin         20592 Jul 24 22:19 init*
-rwxr-xr-x   1 root     root        86020 Jul 28 19:07 ldconfig*
-rwxr-xr-x   1 root     bin          5329 Jul 27 14:10 mkswap*
-rwxr-xr-x   1 root     root         5204 Jul 24 22:20 reboot*
-rwxr-xr-x   1 root     bin         12340 Jul 24 22:20 shutdown*
-rwxr-xr-x   1 root     root         5029 Jul 24 22:20 swapoff*
-rwxr-xr-x   1 root     bin          5029 Jul 24 22:20 swapon*
-rwxr-xr-x   1 root     root        20592 Jul 27 18:18 telinit*
-rwxr-xr-x   1 root     root         7077 Jul 24 22:20 update*
```

```
tmp:
total 0

usr:
total 2
drwxr-xr-x   2 root     root         1024 Jul 29 21:00 adm/
drwxr-xr-x   2 root     root         1024 Jul 29 21:16 lib/

usr/adm:
total 0

usr/lib:
total 0

var:
total 1
*** Un certain nombre d'erreurs apparurent jusqu'a ce que j'inclus
*** ceci et le code de /etc/rc.S pour initialiser /var/run/utmp, mais
*** ceci ne s'appliquera pas forcement a votre systeme.
drwxr-xr-x   2 root     root         1024 Jul 28 19:52 run/

var/run:
total 0
```

B.1.2 Contenu des répertoires du disque utilitaire (ls -lR)

```
total 579
-rwxr-xr-x   1 root     root        42333 Jul 28 19:05 cpio*
-rwxr-xr-x   1 root     root       103560 Jul 29 21:31 elvis*
-rwxr-xr-x   1 root     root        56401 Jul 28 19:06 find*
-rw-r--r--   1 root     root       128254 Jul 28 19:03 ftape.o
-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*
```

B.2 Scripts shell pour construire des disquettes

Ces scripts shell sont fournis uniquement à titre d'exemple. Je les utilise sur mon système pour créer des disquettes de secours. Vous pourrez les trouver utiles, mais dans ce cas, lisez les instructions attentivement - par exemple, si vous spécifiez le mauvais périphérique de pagination, vous retrouverez votre système de fichiers racine méthodiquement et définitivement effacé... faites donc bien attention à la configuration avant de les utiliser !

L'avantage de ces scripts est qu'ils fournissent un moyen rapide de créer un ensemble de disques de secours, en effectuant les taches suivantes :

– copiez un noyau sur un disque d'amorce, et configurez-le grâce à `rdev`, comme expliqué précédemment.

– configurez `mkroot` pour votre système et créez un disque racine. Utilisez les exemples de contenus ci-dessus comme guide pour savoir quoi inclure.

– utilisez `mkutil` pour remplir un ou plusieurs disques utilitaires avec vos programmes favoris.

Il y a deux scripts shell :

– `mkroot` - construit une disquette racine ou amorce/racine.

– `mkutil` - construit une disquette utilitaire.

Tous deux sont actuellement configurés pour être lancés depuis le répertoire père de `boot_disk` et `util_disk`, chacun d'entre eux contenant tout ce qui doit être copié sur sa disquette. Notez que ces scripts shell NE CONFIGURENT PAS automatiquement, ni ne copient tous les fichiers pour vous - vous devez choisir vous-même quels fichiers sont nécessaires, créer les répertoires et copier les fichiers dans ces répertoires. Les scripts shell ne sont que des exemples qui copient le contenu de ces répertoires. Notez aussi que ce sont des scripts primitifs qui ne sont pas destinés à des utilisateurs novices.

Ces scripts contiennent tous deux des variables de configuration au début qui permettent de les configurer facilement pour fonctionner n'importe où. Tout d'abord, configurez les répertoires modèles et copiez dedans tous les fichiers requis. Pour savoir quels répertories et fichiers prendre, jetez un coup d'oeil aux exemples de contenus dans les sections précédentes.

Vérifiez les variables de configuration dans les scripts shell et modifiez les en fonction de vos besoins avant de lancer les scripts.

B.2.1 mkroot - Fabriquer une disquette racine

```
# mkroot : fabrication d'un disque racine - cr\'{e}e une disquette racine
#          en construisant un syst\`{e}me de fichiers dessus, puis la
#          monte et y copie les fichiers n\'{e}cessaires d'apr\`{e}s un mod\`{e}le.
#          Note : le mod\`{e}le d'apr\`{e}s lequel la copie est effectu\'{e}e doit
#          d'abord \^{e}tre mis en place, apr\`{e}s quoi vous pouvez modifier
#          les variables de configuration ci-dessous pour correspondre
#          \`{a} votre syst\`{e}me.
#
# Utilisation : mkroot [ -d swap | ram ]
#       o\`{u} swap demande d'utiliser $SWAPDEV comme p\'{e}riph\'{e}rique de
#       pagination, et ram demande d'utiliser le p\'{e}riph\'{e}rique du
#       disque m\'{e}moire $RAMDISKDEV
```

```
# Copyright (c) Graham Chapman 1996. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Variables de configuration - adaptez les valeurs \`{a} votre syst\`{e}me
#
####  Indiquez le p\'{e}riph\'{e}rique o\`{u} construire le syst\`{e}me de fichiers
####  racines. Un disque m\'{e}moire est plus s\^{u}r - la zone de pagination
####  peut convenir si vous avez beaucoup de m\'{e}moire libre. Si Linux ne
####  peut pas paginer, les choses se g\^{a}tent.
USEDEVICE="ramdisk"             # indiquez "ramdisk" (pour un disque m\'{e}moire)
                                # ou "swap" (pour la zone de pagination)
RAMDISKDEV="/dev/ram"           # p\'{e}riph\'{e}rique du disque m\'{e}moire <==== \`{a}
                                # changer si vous utilisez "ramdisk"
SWAPDEV="/dev/sda7"             # p\'{e}riph\'{e}rique de pagination <==== \`{a}
                                # changer si vous utilisez "swap"
FSBLOCKS=3072                   # taille souhait\'{e}e du syst\`{e}me de fichiers,
                                # en blocs
#
####  Indiquez le r\'{e}pertoire dans lequel vous avez configur\'{e} votre
####  mod\`{e}le de disque racine.
ROOTDISKDIR="./root_disk"       # r\'{e}pertoire du disque racine
MOUNTPOINT="/mnt"               # point de montage temporaire pour la
                                # disquette
DISKETTEDEV="/dev/fd0"          # p\'{e}riph\'{e}rique de la disquette
LOGFL="`pwd`/mkroot.log"        # nom du fichier de log
TEMPROOTFS="/tmp/mkrootfs.gz"   # fichier temporaire pour le syst\`{e}me de
                                # fichiers compress\'{e}
# Fin des variables de configuration

# Variables internes
ROOTDISKDEV=

case $USEDEVICE in
swap|ramdisk)   :;;
*)      echo "Valeur invalide pour la variable USEDEVICE"
        exit;;
esac

clear
echo "    ***************** A T T E N T I O N *****************

Utilisez ce script avec pr\'{e}cautions. Si vous ne comprenez pas son
utilisation, quittez le programme MAINTENANT !"

if [ "$USEDEVICE" = "swap" ]
```

```
then
        ROOTDISKDEV=$SWAPDEV
        echo -e "\nCe script va temporairement retirer la zone de pagination $SWAPDEV"
        echo "et utiliser son espace pour construire un syst\`{e}me de fichiers"
        echo "compress\'{e} \`{a} partir des fichiers situ\'{e}s dans l'arborescence sous"
        echo "$ROOTDISKDIR. Pour que cela soit sans danger, vous devez avoir 8 Mo"
        echo "ou plus de m\'{e}moire, et vous devriez passer dans le mode utilisateur unique"
        echo "gr\^{a}ce \`{a} 'init 1'."
        echo -e "\nSi vous avez utilis\'{e} un disque m\'{e}moire depuis le dernier red\'{e}marrage,"
        echo "red\'{e}marrez MAINTENANT avant d'utiliser ce script."
        echo -e "\nSi ce script \'{e}choue, vous pouvez ne plus avoir de partition de pagination."
        echo "Utilisez 'free' et v\'{e}rifier la taille totale pour voir si elle est"
        echo "correcte. Si la partition de pagination $SWAPDEV manque, effectuez"
        echo "les op\'{e}rations suivantes :"
        echo "  umount $MOUNTPOINT"
        echo "  mkswap $SWAPDEV"
        echo "  swapon $SWAPDEV"
        echo "pour restaurer la partition de pagination $SWAPDEV."
else
        ROOTDISKDEV=$RAMDISKDEV
        echo -e "\nCe script va utiliser un disque m\'{e}moire de $FSBLOCKS ko. Pour"
        echo "le faire en toute s\'{e}curit\'{e}, vous devez avoir au moins 8 Mo de m\'{e}moire."
        echo "Si vous n'avez que 8 Mo de m\'{e}moire, vous devriez vous assurer que rien"
        echo "d'autre ne tourne sur la machine."
        echo -e "\nQuand le script est termin\'{e}, le disque m\'{e}moire sera toujours pr\'{e}sent,"
        echo "et vous devrez donc red\'{e}marrer pour r\'{e}cup\'{e}rer la m\'{e}moire allou\'{e}e au"
        echo "disque m\'{e}moire."
fi

echo -e "
Voulez-vous continuer (o/n)? \c"
read ans
if [ "$ans" != "O" -a $ans != "o" ]
then
        echo "Pas de confirmation - abandon"
        exit
fi

echo "D\'{e}marrage de mkroot \`{a} `date`" > $LOGFL

if [ "$USEDEVICE" = "swap" ]
then
        echo "D\'{e}montage du p\'{e}riph\'{e}rique de pagination $SWAPDEV" | tee -a $LOGFL
        swapoff $SWAPDEV >> $LOGFL 2>&1
fi

echo R\'{e}initialisation du p\'{e}riph\'{e}rique $ROOTDISKDEV" | tee -a $LOGFL
dd if=/dev/zero of=$ROOTDISKDEV bs=1024 count=$FSBLOCKS >> $LOGFL 2>&1
```

```
if [ $? -ne 0 ]
then
        echo "R\'{e}initialisation par dd de $ROOTDISKDEV rat\'{e}e" | tee -a $LOGFL
        exit 1
fi

echo "Cr\'{e}ation du syst\`{e}me de fichiers sur le p\'{e}riph\'{e}rique $ROOTDISKDEV" | tee -a $LOGFL
mke2fs -m0 $ROOTDISKDEV $FSBLOCKS >> $LOGFL 2>&1

echo "Montage du syst\`{e}me de fichiers $ROOTDISKDEV sur $MOUNTPOINT" | tee -a $LOGFL
mount -t ext2 $ROOTDISKDEV $MOUNTPOINT >> $LOGFL 2>&1
if [ $? -ne 0 ]
then
        echo "mount a \'{e}chou\'{e}"
        exit 1
fi

# Copie des r\'{e}pertoires contenant les fichiers
echo "Copie des fichiers depuis $ROOTDISKDIR vers $MOUNTPOINT" | tee -a $LOGFL
currdir=`pwd`
cd $ROOTDISKDIR
find . -print | cpio -dpumv $MOUNTPOINT >> $LOGFL 2>&1
if [ $? -ne 0 ]
then
        echo "cpio a \'{e}chou\'{e}."
        cd $currdir
        exit 1
fi
cd $currdir

fssize=`du -sk $MOUNTPOINT|cut -d"      " -f1`
echo "La taille du syst\`{e}me de fichiers racine non-compress\'{e} est $fssize ko" | tee -a $LOGFL
echo "D\'{e}montage du syst\`{e}me de fichiers de $ROOTDISKDEV" | tee -a $LOGFL
umount $MOUNTPOINT >> $LOGFL 2>&1

echo "Compression du syst\`{e}me de fichiers de $ROOTDISKDEV dans $TEMPROOTFS.
        Cela peut prendre quelques minutes..." | tee -a $LOGFL

#       Pas la peine d'utiliser gzip -9 ici - il prend deux fois plus de
#       temps, et ne sauve que moins d'1 % sur mon disque racine...
dd if=$ROOTDISKDEV bs=1024 count=$FSBLOCKS 2>>$LOGFL | gzip -c > $TEMPROOTFS

fssize=`du -k $TEMPROOTFS|cut -d"       " -f1`
echo "La taille du syst\`{e}me de fichiers racine compress\'{e} est $fssize ko" | tee -a $LOGFL

echo -e "Ins\'{e}rez la disquette dans $DISKETTEDEV et appuyez sur une touche
        ***  Attention : toute donn\'{e}e sur la disquette sera effac\'{e}e !\c"
read ans
```

```
echo "Copie du syst\`{e}me de fichiers compress\'{e} de $TEMPROOTFS vers $DISKETTEDEV" | tee -a $LOGFL
dd if=$TEMPROOTFS of=$DISKETTEDEV >>$LOGFL 2>&1
if [ $? -ne 0 ]
then
        echo "La copie a \'{e}chou\'{e}."
        exit 1
fi

if [ "$USEDEVICE" = "swap" ]
then
        echo "R\'{e}initialisation du p\'{e}riph\'{e}rique de pagination $SWAPDEV" | tee -a $LOGFL
        mkswap $SWAPDEV >> $LOGFL 2>&1
        echo "D\'{e}marrage de la pagination sur le p\'{e}riph\'{e}rique $SWAPDEV" | tee -a $LOGFL
        swapon $SWAPDEV >> $LOGFL 2>&1
fi

echo "Destruction de $TEMPROOTFS" | tee -a $LOGFL
rm $TEMPROOTFS

echo "mkroot termin\'{e} \`{a} `date`" >> $LOGFL

echo "Cr\'{e}ation de la disquette racine termin\'{e}e - lisez le fichier de log $LOGFL"
```

B.2.2 mkutil - Fabriquer une disquette utilitaire

```
# mkutil : fabrication d'une disquette utilitaire - cr\'{e}e une disquette
#          utilitaire en construisant un syst\`{e}me de fichiers dessus,
#          puis en le montant et en copiant dessus les fichiers requis
#          d'apr\`{e}s un mod\`{e}le.
#          Note : le mod\`{e}le d'apr\`{e}s lequel sont copi\'{e}s les fichiers
#          doit \^{e}tre configur\'{e} d'abord, apr\`{e}s quoi vous pouvez changer
#          les variables de configuration ci-dessous pour correspondre
#          \`{a} votre syst\`{e}me.

# Copyright (c) Graham Chapman 1996. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Variables de configuration ...
UTILDISKDIR=./util_disk         # nom du r\'{e}pertoire contenant le mod\`{e}le
MOUNTPOINT=/mnt                 # point de montage temporaire pour la
                                # disquette
DISKETTEDEV=/dev/fd0            # nom du p\'{e}riph\'{e}rique du lecteur de disquette

echo $0: cr\'{e}ation d'une disquette utilitaire
```

```
echo Attention : toute donn\'{e}e pr\'{e}sente sur la disquette sera \'{e}cras\'{e}e !
echo Ins\'{e}rez une disquette dans $DISKETTEDEV et appuyez sur une touche ...
read anything

mke2fs $DISKETTEDEV
if [ $? -ne 0 ]
then
        echo mke2fs a \'{e}chou\'{e}
        exit
fi

# Tout type de syst\`{e}me de fichiers peut convenir ici
mount -t ext2 $DISKETTEDEV $MOUNTPOINT
if [ $? -ne 0 ]
then
        echo mount a \'{e}chou\'{e}
        exit
fi

# Copie des r\'{e}pertoires contenant les fichiers
cp -dpr $UTILDISKDIR/* $MOUNTPOINT

umount $MOUNTPOINT

echo Disquette utilitaire cr\'{e}\'{e}e
```