Petit guide pratique d'installation de la Glibc

Adaptation française du *Glibc-Install-HOWTO*

Kai Schlachter `<linux CHEZ murphyslantech POINT de>`

Petit guide pratique d'installation de la Glibc: Adaptation française du *Glibc-Install-HOWTO*

par Kai Schlachter
Adaptation française: Nicolas Jadot
Préparation de la publication de la v.f.: Jean-Philippe Guérard

Version :1.01.fr.1.0

Date de publication 25 janvier 2005

Table des matières

Chapitre 1. Préface

1. Copyright

2. Historique

- v1.01 : Correction de citations mal placées

3. Remerciements

Je tiens à remercier tout spécialement Nico Schmoigl qui m'a aidé à remonter mon système après un crash dû à l'utilisation de la commande **make install** pour glibc alors que mon système était en plein fonctionnement. C'est la raison principale qui m'a conduit à écrire ce petit guide pratique.

Je remercie également toutes les personnes qui m'ont aidé tout du long de ce travail ; utiliser le bon format de guide, trouver les bonnes typographies, etc.

Chapitre 2. Introduction

J'expliquerai dans ce guide pratique comment installer une nouvelle version de la bibliothèque glibc™ sur votre système.

J'ai écris ce manuel afin de protéger les autres utilisateurs des problèmes que j'ai rencontrés.

Ce document est une sorte de ligne de vie, faite des paramétrages et des méthodes qui ont fonctionné dans mon cas. L'auteur décline toute responsabilité quant au contenu de ce document. Utilisez les concepts, exemples et autres à vos risques et périls. Je vous recommande cependant vivement de sauvegarder l'intégralité de votre système avant toute installation majeure et de faire des sauvegardes régulières.

Si vous avez la moindre suggestion, ou si vous trouvez une autre erreur dans une distribution, avec le moyen de la *fixer*, faites le moi savoir, en m'écrivant à : `<linux CHEZ murphyslantech POINT de>`.

1. Pourquoi ?

C'est effectivement une bonne question. Pour quelles raisons peut-on avoir à installer une nouvelle version de glibc ? Pour plusieurs raisons :

- Vous êtes développeur et avez besoin des nouvelles fonctions de la bibliothèque ;
- Vous voulez compiler un programme qui nécessite la nouvelle bibliothèque ;
- Vous aimez le frisson des erreurs et bugs des nouvelles versions ;-).

2. Quoi ?

Si vous ne savez pas ce qu'est vraiment glibc, ne vous inquiétez pas ! Quand j'ai eu mes premiers soucis avec un nouveau programme que je voulais compiler, je savais simplement que ma version de glibc n'était pas suffisante pour la compilation. Maintenant que j'en sais davantage, je vais essayer d'expliquer de manière simple ce à quoi sert glibc.

Le paquetage glibc contient une bibliothèque écrite en langage C. Les bibliothèques sont des composants très utiles en programmation ; plutôt que de réinventer la roue à partir du néant pour des opérations comme le calcul de la racine carrée d'un nombre, ce type de fonction est stocké dans des fichiers séparés — les bibliothèques. Quand une nouvelle version d'une bibliothèque est publiée, elle contient souvent de nouvelles fonctions, utilise des algorithmes plus performants pour celles déjà existantes, et ainsi de suite.

C'est pourquoi certains programmes se plaignent de versions trop anciennes de glibc : la version courante ne contient tout simplement pas la fonction dont le programme a besoin pour son exécution.

Je sais que cette explication n'est pas techniquement correcte dans tous les détails, mais elle donne une explication simple de l'architecture sous-jacente.

Chapitre 3. Préparatifs

L'installation de glibc n'est pas une tâche facile. Il vous faudra effectuer un certain nombre d'opérations préalables, spécialement pour le cas où quelque chose tournerait mal. C'est ce qui m'est arrivé lors de ma première installation de glibc, et je n'avais fait aucun de ces préparatifs.

1. Ce qu'il vous faut

Schématiquement, il vous faut deux choses : les logiciels dont fonctionnent déjà sur votre machine (soit, les paquetages de votre distribution) et les paquetages source de différents programmes.

1.1. Les logiciels dont vous aurez besoin

- un compilateur gcc™ opérationnel ;
- une ancienne version de glibc™ ;-) ;
- les binutils GNU™ ;
- le programme GNU make™ ;
- le core-utils GNU™ ;
- le programme GNU tar™ ;
- le shellbash™ ou tout autre de votre choix ;
- très pratique mais pas le mieux : Midnight Commander™ ;
- votre éditeur favori (vi™, jed™, et cætera).

1.2. Les codes sources qu'il vous faut

- bash™ ou celui de votre shell ;
- GNU tar™ ;
- GNU core-utils™ ;
- GNU make™ ;
- bien entendu : glibc™ ;
- éventuellement : gcc™.

2. Choses spéciales à faire

Comme vous vous préparez à remplacer la bibliothèque de base sur laquelle de nombreux programmes reposent, vous pouvez aisément imaginer que divers problèmes peuvent survenir.

Dans mon cas, tout fonctionnait jusqu'au moment où j'ai tapé **make install**. A la moitié du processus d'installation, j'ai reçu un message d'erreur m'indiquant que la commande **rm** ne pouvait fonctionner, et j'ai découvert que toutes les commandes comme **cp**, **ls**, **mv**, **ln**, **tar**, etc ne fonctionnaient plus ; toutes me répondaient qu'elles ne trouvaient plus les parties de la bibliothèque qui leur étaient nécessaires (NDT : j'ai eu aussi ce *petit* soucis, toutefois dans un cas plus trivial ;-)).

Mais il existe une solution. Vous pouvez, lors de la compilation d'un programme, forcer l'inclusion des fonctions de la bibliothèque à l'intérieur du programme, de telle sorte qu'il n'ait plus à les rechercher à l'intérieur.

Pour cette raison, dans ce chapitre, nous allons compiler des versions statiques de tous les utilitaires dont nous avons besoin.

2.1. Les choses indispensables

2.1.1. GNU binutils

1. Chargez la toute dernière version depuis ftp.gnu.org/gnu/binutils ; à la date de rédaction de ce guide, le numéro de la version la plus récente était 2.14.

2. Ouvrez le paquetage :

   ```
   tar xIvf
   binutils-2.14.tar.bz2
   ```

   .

3. Changez de répertoire :

   ```
   cd binutils-2.14
   ```

   .

4. Configurez les *makefiles* :

   ```
   ./configure
   ```

   .

5. Compilez :

   ```
   make
   ```

   .

6. Installez avec :

   ```
   make install
   ```

   .

Si quelque chose se déroule mal durant la compilation de binutils, en rapport avec gettext (indiqué par des erreurs comme : « référence à lib_intl non déclarée » ou quelque chose d'approchant) installez la nouvelle version, disponible à ftp.gnu.org/gnu/gettext.

Si cela ne suffit pas, essayez de désactiver le support du langage natif en utilisant :

```
./configure
--no-nls
```

.

Il n'est pas nécessaire de compiler une version statique de binutils, bien que cela ne cause aucun problème, mais j'ai rencontré de nombreux systèmes fonctionnant avec de très vieilles versions et eu des erreurs presque à chaque fois aussi je pense qu'il est utile de le mentionner ici.

2.1.2. GNU make

La commande **make** règle l'ensemble de la compilation des sources, appelant gcc et tous les autres programmes nécessaire à la compilation. Comme il est possible que vous soyez contraint à compiler quelque chose si une erreur survient avec votre nouvelle glibc, c'est une bonne idée que **make** soit compilé de manière statique ; dans le cas contraire, il pourrait ne pas fonctionner si une erreur survient.

1. Téléchargez les sources depuis : ftp.gnu.org/gnu/make/ ; à la date de rédaction, la version est 3.80.
2. Déballez les sources :

   ```
   tar xIvf make-3.80.tar.bz2
   ```

   .
3. Changez de répertoire :

   ```
   cd make-3.80
   ```

   .
4. Prenez garde à compiler de manière statique :

   ```
   export CFLAGS="-static -O2 -g"
   ```

   .
5. Lancez le script de configuration :

   ```
   ./configure
   ```
6. Compilez :

   ```
   make
   ```

   .
7. Installez les binaires:

   ```
   make install
   ```

   .
8. Vérifiez :

   ```
   make -v
   ```

   Vous devez maintenant voir la dernière version installée. Sinon, recherchez les anciens fichiers binaires et remplacez les par des liens symboliques vers les nouvelles versions.

Félicitations ! Vous avez compilé un autre programme statique.

2.1.3. GNU core-utils

core-utils contient des commandes comme : **cp**, **rm**, **ln**, **mv**, etc. En cas d'erreur durant l'installation, c'est une nécessité absolue pour pouvoir espérer remonter le système, aussi des binaires statiques sont vraiment nécessaires ici.

1. De nouveau, téléchargez les sources depuis ftp.gnu.org/gnu/coreutils/ ; au moment de la rédaction, la version courante est la 5.0.
2. Dépaquetez les sources :

   ```
   tar xIvf
   coreutils-5.0.tar.bz2
   ```
3. Changez de répertoire :

```
cd
coreutils-5.0
```

4. Prenez garde au fait que les binaires doivent être liés statiquement :

   ```
   export CFLAGS="-static -O2 -g"
   ```

5. Configurez le paquetage :

   ```
   ./configure
   ```

6. Compilez les binaires :

   ```
   make
   ```

7. Et installez les :

   ```
   make
   install
   ```

8. Vérifiez que la bonne version de core-utils est utilisée :

   ```
   cp --version
   ```

   . Vous devez obtenir un numéro de version correct, dans le cas contraire, supprimez les anciens binaires et remplacez les par des liens symboliques vers les nouveaux.

Maintenant que les binaires de ces outils de base sont des versions statiques, vous êtes sûr qu'il fonctionneront en cas de besoin.

2.1.4. GNU tar

Vous avez déjà utilisé GNU tar pour dépaqueter les programmes compilés et installés. Mais vous avez besoin de compiler un programme requis par glibc après un crash et, dans cette situation (expérience vécue) il est particulièrement utile de disposer d'un **tar** opérationnel pour dépaqueter les programmes manquants. Il nous faut également, avec tar, prendre garde à la compression bz2, qui n'est pas incluse dans la distribution normale des sources de tar.

1. Récupérez les sources de GNU tar sur ftp.gnu.org/gnu/tar ; à la date de rédaction, la dernière version est la 1.13.

2. Comme de multiples archives tar sont compressées avec bzip2, nous préférerons disposer du support intégré, plutôt que de devoir travailler avec des *pipes* ; il nous faut donc récupérer le patch à ftp://infogroep.be/pub/linux/lfs/lfs-packages/4.1/tar-1.13.patch.

3. Dépaquetez les sources :

   ```
   tar xzvf tar-1.13.tar.gz
   ```

4. Copiez le patch dans le répertoire contenant les sources de tar

   ```
   cp tar-1.13.patch tar-1.13/
   ```

5. Appliquez le :

   ```
   patch -Np1 -i tar-1.13.patch
   ```

6. Posez les options du compilateur pour obtenir un binaire statique :

   ```
   export CFLAGS="-static -O2 -g"
   ```

7. Nous sommes maintenant prêt pour la configuration :

   ```
   ./configure
   ```

8. La compilation :

   ```
   make
   ```

9. Et enfin l'installation :

   ```
   make
   install
   ```

10. Faites une dernière vérification pour vous assurer que la nouvelle version sera utilisée à partir de maintenant :

    ```
    tar
    --version
    ```

    . La version que vous venez d'installer doit apparaître, sinon, remplacez les anciens binaires par des liens symboliques vers les nouveaux.

Si vous avez des problèmes dans l'exécution de la commande **make**, essayez de désactiver le support du langage natif (*nls : native-language support*). Pour ce faire, utilisez la commande configure avec l'option

```
--disable-nls
```

Note : dans la nouvelle version de tar, vous devez utiliser l'option `-j` pour décompresser les fichiers .bzip2, aussi plutôt que

```
tar xIvf
anyfile.tar.bz2
```

vous devez maintenant utiliser

```
tar
xjvf anyfile.tar.bz2
```

. J'ignore les raisons de ce changement, mais ça fonctionne bien.

2.1.5. Le shell bash

Je préfère utiliser le shell bash ; si vous en utilisez un différent, assurez-vous d'avoir installé une version statique de celui-ci avant d'installer glibc.

1. Récupérez bash à : ftp.gnu.org/gnu/bash/. Téléchargez la version la plus récente que vous trouverez ; à la date de rédaction, celle-ci est la 2.05b.

2. Dépaquetez les sources :

   ```
   tar xzvf
   bash-2.05b.tar.gz
   ```

   créera un répertoire nommé `bash-2.05b` contenant toutes les sources.

3. Déplacez-vous dans ce répertoire :

   ```
   cd
   bash-2.05a
   ```

4. Configurez pour obtenir une version statique :

```
export CFLAGS="-static -O2 -g"
```

5. Configurez le makefile :

   ```
   ./configure
   ```

   . Si vous désirez que votre bash possède certaine fonctionnalités spéciales, reportez-vous à la commande

   ```
   ./configure --help
   ```

   pour la liste des options.

6. Compilez le tout :

   ```
   make
   ```

7. Installez les binaires :

   ```
   make
   install
   ```

   . Ils seront installés dans le répertoire `/usr/local/bin/`.

8. Vérifiez qu'une autre version ne reste pas tapie quelque part (comme dans ma Suse : `/bin/`), en copiant le fichier :

   ```
   cp /usr/local/bin/bash
   /bin/
   ```

   . Nous n'utilisons pas ici de lien symbolique car, tant lors du démarrage de la machine qu'au lancement de bash, des problèmes peuvent survenir avec les liens symboliques.

Vous avez maintenant installé une version statique de bash. C'est pour cette raison que le binaire est beaucoup plus volumineux qu'à l'accoutumée, mais il fonctionnera quelles que soient les circonstances.

Si vous préférez utiliser un autre shell, vous êtes libre de le faire, mais assurez-vous qu'il s'agit d'une version liée statiquement. N'hésitez pas à me communiquer la méthode pour construire une version statique du shell de votre choix ; il y a de grandes chances qu'elle soit reprise dans la prochaine révision de ce document.

2.2. Logiciels divers

2.2.1. Midnight Commander

Midnight Commander est un gestionnaire de fichiers très utile, apportant de nombreuses fonctionnalités intéressantes, comme la décompression directe des fichiers compressés, la copie, le déplacement de fichiers et d'autres commandes intégrées, ainsi qu'un éditeur.

Pour compiler ce logiciel, glib doit être installé ; cela est le cas dans certaines distributions. Si vous rencontrez une erreur lors de l'exécution de la commande **make** vous indiquant que ld ne peut pas trouver glib, vous devrez installer cette bibliothèque avant tout autre chose. Vous pouvez en trouver les sources à : ftp.gnome.org/pub/gnome/sources/glib/2.2/, et l'installation est directe .

Les étapes de la construction de Midnight Commander sont :

1. Récupérer les sources depuis http://www.ibiblio.org/pub/Linux/utils/file/managers/mc/ ; à la date de rédaction, la version la plus récente est la 4.6.0.

2. Dépaqueter les sources :

```
tar xzvf mc-4.6.0.tar.gz
```

3. Se déplacer dans le répertoire nouvellement créé :

```
cd mc-4.6.0
```

4. Configurer :

```
./configure
```

5. Compiler :

```
make
```

6. Tout installer :

```
make install
```

Chapitre 4. Installation de glibc en soi

Maintenant nous atteignons le point crucial : l'installation de glibc.

1. Récupérer et installer les sources

Plusieurs versions de glibc sont disponibles, mais les nouvelles versions ne sont pas préférables aux plus anciennes dans tous les cas. La meilleure chose à faire pour savoir lesquelles fonctionnent et lesquelles vous ne devez pas utiliser est de se renseigner sur les différents groupes de discussion sur Internet. Si vous connaissez quelqu'un à qui demander, parlez lui en. Peut-être a-t'il déjà installé la nouvelle version et peut il vous dire que la version x.y.z est une PITA , mais que la version a.b.c fonctionne vraiment bien !

J'ai décidé d'installer la glibc-2.2.4, puisque l'on m'avait dit qu'elle fonctionne bien, mais le choix de la version demeure de votre ressort.

Bien, maintenant, au travail :

1. Récupérez les sources sur ftp.gnu.org/gnu/glibc/ ; comme dit plus haut, j'ai utilisé la version 2.2.4.

2. Dépaquetez les sources :

   ```
   tar -xzvf
   glibc-2.2.4.tar.gz
   ```

3. Vous aurez besoin, en plus, du paquetage « linuxthreads » que l'on peut trouver dans le répertoire `linuxthreads` sur ftp.gnu.org. Le fichier est :

   ```
   glibc-linuxthreads-2.2.4.tar.gz
   ```

   . Assurez-vous que le numéro de version correspond à celui de votre arborescence des sources de glibc.

4. Copiez le paquetage linuxthreads dans le répertoire contenant les sources de glibc :

   ```
   cp
   glibc-linuxthreads-2.2.4.tar.gz glibc-2.2.4
   ```

5. Déplacez vous dans ce répertoire :

   ```
   cd
   glibc-2.2.4
   ```

6. Dépaquetez linuxthreads :

   ```
   tar xzvf
   linux-threads-2.2.4.tar.gz
   ```

7. Configurez le paquetage :

   ```
   ./configure
   --enable-add-ons=linuxthreads
   ```

   . Cela configurera le paquetage de telle sorte que linuxthreads soit inclus dans la compilation ; ceci est nécessaire pour la compatibilité avec d'autres systèmes Linux. Par exemple, les programmes que vous compilerez sur votre machine ne fonctionneront probablement pas sur d'autres machines si vous omettez d'inclure ce paquetage.

8. Enfin, lancez la compilation de glibc :

```
make
```

. Cela peut prendre un certain temps (à peu près une demi-heure sur le Duron XP 1,5 GHz de l'auteur).

Maintenant que la bibliothèque est compilée, tout est prêt pour l'installation, mais les choses ne seront pas si simple cette fois.

2. Installation

Pour installer glibc, il vous faut un système où rien ne s'exécute, car de nombreux processus (par exemple sendmail) utilisent continuellement les services de la bibliothèque et donc, interdisent le remplacement du fichier. Il vous faut donc un système « nu », n'exécutant rien d'autre que le strict nécessaire. Vous pouvez atteindre cet objectif en passant l'option de démarrage `init=/bin/bash` au noyau. En fonction de votre gestionnaire d'amorçage, il vous faudra peut-être faire d'autres choses. Ci-après, nous expliquerons ce qu'il vous faut faire en prenant comme exemples les deux gestionnaires d'amorçage les plus courants, LILO (LInux-LOader) et GNU grub.

2.1. LILO

Pour lancer un système « minimum », relancez l'ordinateur et, à l'invite de LILO, entrez le nom du noyau que vous désirez lancez et, à la suite, ajoutez

```
init=/bin/bash
```

avant de presser la touche **Entrée**. Si vous envisagez de remplacer souvent votre glibc, ce peut être une bonne idée d'inclure une configuration spécifique dans votre fichier `/etc/lilo.conf`. Pour plus de détails, reportez-vous à la page de manuel de LILO.

2.2. Grub

Grub est un gestionnaire d'amorçage plus récent, avec un support étendu de différents systèmes d'exploitation et systèmes de fichiers (par exemple, il supporte le démarrage depuis des partitions reiserfs). Si vous désirez en savoir plus, reportez vous à http://www.gnu.org/software/grub/, où vous trouverez tout le nécessaire.

Si grub est déjà installé chez vous, vous utilisez probablement l'interface en mode texte pour sélectionner le noyau que vous voulez lancer. Grub dispose d'une intéressante fonctionnalité — au lieu de tout refaire à la main, vous pouvez simplement sélectionner l'entrée qui vous intéresse et, alors, taper **e**, ce qui provoquera l'affichage d'un menu optionnel. Dans ce menu, vous verrez la commande exécutée par grub avant le lancement du noyau. Sélectionnez la ligne indiquant

```
kernel="/où/est-le-noyau-et-quelles-sont-les-options"
```

et tapez **e** de nouveau. Maintenant, vous pouvez éditer cette ligne. Vous ajoutez simplement

```
init=/bin/bash
```

et tapez **Entrée** pour rendre les modifications effectives, tapez **b** pour démarrer.

3. Après le démarrage du noyau…

… Vous allez vous retrouver dans un environnement shell absolument minimal.

Il ne vous sera même pas demandé de mot de passe ! A ce moment, vous êtes le super-utilisateur absolu ; personne ne peut se connecter car le système est en mode mono-utilisateur, aussi faites très attention à ce que vous faites. Aucun droit n'est posé sur les fichiers ni rien d'autre !

L'invite ressemblera sûrement à

```
init-x.y#
```

. La racine (/) ayant été montée en lecture seule, il vous sera impossible d'enregistrer la nouvelle librairie sur le disque dur. Pour rendre la racine accessible en lecture/écriture, entrez la commande :

```
mount -o remount,rw /
```

. Si les sources se trouvent sur une autre partition, vous devez également la monter (dans le cas de l'auteur, cela signifie monter son système raid) :

```
mount -t
reiserfs /dev/md0 /usr/src
```

. Comme vous le voyez, on définit également le type du système de fichier car **mount** n'ira rien chercher dans dans `/etc/fstab`.

Maintenant, vous pouvez aller dans le répertoire contenant les sources et taper :

```
make install
```

.

Si vous le désirez, c'est maintenant le bon moment pour prier pour que tout se passe bien… ;-)

Si tout se passe parfaitement, vous retrouverez l'invite de commande après l'installation sans aucun message d'erreur. Dans tous les autres cas, reportez vous à Chapitre 5, *En cas de problème — si quelque chose a mal tourné…* .

Si tout s'est bien passé, lancez

```
ldconfig -v
```

pour mettre à jour le cache des bibliothèques.

Félicitations ! La bibliothèque est installée avec succès. Maintenant, tapez : **mount -o remount,ro /** pour être sûr que toutes les données sont écrites sur le disque dur.

Lancez la procédure de redémarrage :

```
exit
```

. Cela provoquera un message d'erreur indiquant que vous avez causé un kernel-panic. Si possible, relancez l'ordinateur par **CTRL+ALT+DEL**, sinon utilisez le bouton reset.

Essayez de lancer votre noyau habituel. Si tout se passe bien, vous êtes prêt à utiliser votre nouvelle bibliothèque.

Chapitre 5. En cas de problème — si quelque chose a mal tourné…

Si vous arrivez à cette section en ayant suivi toutes les instructions fournies plus haut, vous avez probablement été confronté au problème inhérent à la multiplicité des distributions Linux, certaines ne placent pas les choses où elles devraient être mais ailleurs. Les distributions Suse sont les plus fameuses de celles-là, mais d'autres peuvent avoir ce problème. Si vous rencontrez ce problème et trouvez la cause de l'erreur — et j'espère la solution — soyez gentil de me le faire savoir, afin que je l'ajoute à ce document.

Je pense que cette section ne sera jamais vraiment complète, mais je vais tenter de lister un certain nombre d'erreurs possibles et les solutions pour s'en sortir.

1. Erreurs avec les commandes configure ou make durant la compilation de glibc

Vous pouvez parfois obtenir des messages d'erreur durant la configuration vous indiquant que, par exemple, une dépendance n'est pas satisfaite, typiquement en ce qui concerne les logiciels ou les paquetages de bibliothèques trop anciens. J'ai rencontré ce type de problème avec une série de programmes, tout spécialement durant la compilation des versions statiques des différents outils nécessaires. La solution devrait normalement être très simple : récupérer les versions à jour des programmes ou des bibliothèques incriminées et les compiler en se conformant ou instructions fournies (habituellement dans le fichier `README`, `INSTALL` ou tout autre approchant).

Mais il est quelques cas où cela ne voudra pas fonctionner. Par exemple, j'ai eu des soucis pour compiler une nouvelle version de binutils (c'est une des raisons pour lesquelles je le mentionne dans les pré-requis), alors que j'en avais besoin pour compiler glibc. Le script configure me retournait une erreur m'indiquant « Votre glibc est trop ancienne ! » Aussi j'ai pensé, *Ici, le serpent commence à se mordre la queue*. Il existe heureusement une solution à ce problème : si vous ne pouvez pas faire un grand pas en avant, essayer d'en faire plusieurs petits, mais davantage.

Dans ma distribution, je disposais de la glibc version 2.1.1. Pour remédier à l'erreur, j'ai tenté la compilation de la version 2.1.3, sans problème. Après avoir installé cette version, j'ai retenté la compilation de binutils, et toutes les dépendances ont été, cette fois, résolues.

Si vous rencontrez cette sorte de « boucle » essayez de rechercher la version minimal du logiciel requises, et téléchargez la (je pense que c'est une des raisons pour laquelle autant d'anciennes versions demeurent sur les serveurs ftp). Après avoir mené à bien la compilation et l'installation, essayez de construire le logiciel réticent ; dans la plupart des cas cela devrait aboutir. Il sera peut-être nécessaire d'utiliser de nouveau cette méthode pour compiler des logiciels anciens ou manquants. C'est ce que j'appelle « la longue queue du rat » ou « l'effet domino ». Vous ne vouliez faire qu'une chose, mais vous êtes contraints d'en effectuer plusieurs autres avant de pouvoir faire aboutir la première. Cela peut être ennuyeux, mais il y a un point positif à cela : après, vous êtes à peu près certains que beaucoup des logiciels réellement anciens seront remplacés quand vous terminerez l'installation.

2. Quelque chose se passe mal durant l'installation (make install)

L'erreur la plus commune est de ne pas disposer de l'ensemble des outils en version statique dans ce cas, vous ne pouvez utiliser que la commande **cd** et rien d'autre. C'est pour cette raison que j'ai décrit en détail, dans ce guide pratique, comment recréer des versions statiques de tous les outils nécessaires.

Le seul outil non statique est **mount** et, pour de bonnes raisons à mon avis, il est inclus dans le paquetage linux-utils, qui contient également **login**, **passwd**, etc. De même que vous ne pouvez utiliser de version compilée statiquement en association avec PAM ou d'autres logiciels de sécurité, ce serait une erreur de les recompiler de manière statique quelles que soient les circonstances. Vous êtes bien entendu libre de le faire si vous êtes certain de ce que vous faites.

2.1. Retrouver une configuration opérationnelle

La méthode permettant de retrouver une configuration en état de marche est très simple si vous utilisez des outils compilés de manière statique : allez dans le répertoire `/usr/local/lib/` et déplacez les fichiers nouvellement installés dans un autre répertoire (par exemple, `/usr/local/lib/deplace/`). Vous pourrez les identifier grâce à leurs numéros de versions, qui devrait être identique que celui de la glibc (dans mon cas, les noms de fichiers répondaient tous au schéma lib*-2.2.4), et bien entendu par les dates et heures de création. Il est assez rare que deux bibliothèques différentes aient le même numéro de version au même moment — je ne l'ai moi-même jamais vécu — mais pour vous assurer quand même de ne rien détruire d'important pour votre système vérifiez les date et heure de création. S'il est installé, Midnight Commander est très utile pour cel.

Vous pouvez essayer d'effacer les fichiers `ld-2.2.4.so` et `libc-2.2.4.so` et de lancer **ldconfig -v** ensuite, avant de détruire tous les fichiers endommagés. Cela vous permettra au moins d'utiliser la plupart des programmes et, dans tous les cas, vous pourrez utiliser Midnight Commander.

N'oubliez pas non plus de lancer au moins une fois **ldconfig -v** après avoir détruit les fichiers incriminés.

2.1.1. Remédier aux causes du plantage système

Une source courante de problème est que votre distribution stocke les fichiers des bibliothèques à un emplacement différent que celui utilisé par les routines d'installation que vous avez lancées, donc il arrive souvent que deux versions différentes soient simultanément en fonction, se perturbant l'une l'autre. Dans mon cas, j'ai eu beaucoup de soucis à cause d'une seconde copie de `libc6.so` dans le répertoire `/lib`, un lien symbolique entre ce fichier et son pendant contenu dans le répertoire `/usr/local/lib` a remédié au problème.