HOWTO Glibc 2

Eric Green, ejg3@cornell.edu v1.5, 8 février 1998

(Maintenance de la version française par Géraud Canet, canet@labri.u-bordeaux.fr) Le HOWTO Glibc 2 couvre l'installation et l'utilisation de la bibliothèque C GNU version 2 (libc 6) sur les systèmes Linux.

Contents

1 Introduction

1.1 Note du traducteur

Ce document a été traduit et maintenu jusqu'à la version 1.4 par Olivier Tharan, dont je ne fais que reprendre modestement le travail.

Le document est truffé de signes "cabalistiques" comme des dièse et des pourcent, dont la représentation en SGML est respectivement `#` et `%` ; le source LaTeX généré par les outils de conversion SGML introduit le signe antislash devant ces signes et reste dans la version PostScript de ce document. Les données d'exemple sont donc erronées, mais vous pouvez vous rapporter à la version HTML du document qui n'a pas ces erreurs. La version d'origine connait les mêmes problèmes.

1.2 À propos de glibc 2

Glibc 2 est la toute dernière version de la bibliothèque C du GNU. Elle fonctionne pour l'instant sans modifications sur les systèmes GNU Hurd, et les systèmes Linux sur architectures i386, m68k et alpha. Les adaptations pour Linux PowerPC, MIPS, Sparc, Sparc 64 et ARM seront dans la version 2.1. À l'avenir, d'autres architectures et systèmes d'exploitation seront supportés.

Sur Linux, glibc 2 est utilisée comme libc avec un numéro majeur de version égal à 6, le successeur de la libc 5 pour Linux. Elle est destinée par les développeurs de la libc Linux à remplacer en fin de compte la libc 5. À l'heure de la version 2.0.6, on considère que glibc est de qualité suffisante pour être utilisée en production. La version 2.1 (à venir dans un futur proche) sera prête pour une utilisation normale avec l'ajout de plus de portages et de possibilités.

Il y a trois extensions disponibles en option sur la glibc 2 :

Crypt

Le paquetage UFC-crypt pour le cryptage des données. Il est disponible séparément à cause de restrictions à l'exportation.

LinuxThreads

Une mise en oeuvre de l'interface Posix 1003.1c `"pthread"`.

Locale data

Contient les données nécessaires à la construction des fichiers de données locale pour utiliser les possibilités d'internationalisation de la glibc.

Les extensions crypt et LinuxThreads sont fortement recommandées... Ne pas les utiliser risque de les rendre incompatibles avec les bibliothèques d'autres systèmes. (Si vous ne voulez pas les utiliser, vous devez ajouter l'option –disable-sanity-checks quand vous lancez configure.)

1.3 À propos de ce document

Ce HOWTO couvre l'installation de la bibliothèque glibc 2 sur un système Linux existant. Il est fait pour les utilisateurs de systèmes à base de processeurs Intel qui utilisent pour l'instant la libc 5, mais les utilisateurs d'autres systèmes et de bibliothèques similaires (comme la glibc 1) devraient pouvoir utiliser ces informations en substituant les noms de fichiers et d'architecture adéquats aux endroits appropriés.

La copie la plus récente de ce HOWTO fait partie du *Linux Documentation Project* `http://sunsite.unc.edu/LDP` , ou bien vous en trouverez une version à `http://www.imaxx.net/~thrytis/glibc/Glibc2-HOWTO.html` .

1.4 Changements récents dans ce document

Différences entre la version 1.5 et 1.4 :

- Indexage ajouté par Ed Bailey.
- Changement de mon adresse électronique.

Différences entre la version 1.4 et 1.3 :

- Changé l'état courant d'expérimental à production.
- Mis à jour la liste des portages en développement.
- Mis à jour la dernière version en 2.0.6.

2 Choix de la méthode d'installation

Il y a plusieurs manières d'installer glibc. Vous pouvez installer les bibliothèques pour les tester, en continuant d'utiliser les bibliothèques existantes par défaut, tout en vous permettant d'essayer les nouvelles bibliothèques par l'utilisation d'options différentes à la compilation de votre programme. L'installation de cette manière facilite aussi la désinstallation de glibc dans le futur (cependant, tout programme lié avec la glibc ne fonctionnera plus une fois que vous aurez enlevé les bibliothèques). L'utilisation de glibc comme une bibliothèque de test demande que vous compiliez les bibliothèques à partir des sources. Il n'y a pas de distribution binaire pour l'installation de cette manière. Cette installation est décrite dans la section 4 (installation comme bibliothèque de test).

L'autre manière de l'installer, décrite dans ce document, est d'utiliser glibc comme bibliothèque principale. Tous les nouveaux programmes que vous compilerez sur votre système utiliseront glibc, bien que vous puissiez toujours lier des programmes à vos anciennes bibliothèques par des options de compilation différentes. Vous pouvez, soit installer les bibliothèques à partir de binaires, soit compiler la bibliothèque vous-même. Si vous voulez changer les options d'optimisation ou de configuration, ou utiliser une extension qui n'est pas distribuée en paquetage binaire, vous devrez obtenir la distribution de sources et compiler vous-même. Cette procédure d'installation est décrite dans la section 5 (installation comme bibliothèque C principale).

Frodo Looijaard décrit encore une autre manière d'installer glibc. Sa méthode implique l'installation de glibc comme bibliothèque secondaire et la mise en place d'un compilateur croisé (*cross-compiler*) pour compiler en utilisant glibc. Pour cette méthode, la procédure d'installation est plus compliquée que l'installation en tant que bibliothèque de test décrite dans ce document, mais facilite la compilation et l'édition de liens avec glibc. Cette méthode est décrite dans son document *Installer glibc-2 sur Linux* `http://huizen.dds.nl/~frodol/glibc/` (en anglais).

Si vous utilisez en ce moment un système Debian 1.3 et ne voulez pas le mettre à jour en installant la version instable de Debian pour utiliser glibc, le *HOWTO libc5 vers libc6 Debian* `http://www.gate.net/~storm/FAQ/libc5-libc6-Mini-HOWTO.html` décrit la manière d'utiliser les paquetages Debian pour mettre votre système à jour.

Si vous installez glibc 2 sur un système important, vous voudrez sans doute utiliser l'installation de test. Même s'il n'y a pas de bogues, certains programmes devront être modifiés avant compilation à cause de changements dans les prototypes de fonctions et les types de données.

3 Obtenir la bibliothèque

La glibc 2 contient le paquetage glibc et trois paquetages supplémentaires optionnels, LinuxThreads, Locale et Crypt. Vous pouvez trouver les sources à

- `ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.6.tar.gz`
- `ftp://prep.ai.mit.edu/pub/gnu/glibc-linuxthreads-2.0.6.tar.gz`
- `ftp://prep.ai.mit.edu/pub/gnu/glibc-localedata-2.0.6.tar.gz`
- `ftp://prep.ai.mit.edu/pub/gnu/glibc-crypt-2.0.6.tar.gz`

Il vous faudra à peu près 150 Mo d'espace disque pour la compilation complète et l'installation. L'installation binaire de base du paquetage de la bibliothèque prend aux alentours de 50 Mo.

Les paquetages binaires pour la version 2.0.6 ne sont pas disponibles. Les paquetages binaires pour la version 2.0.4 sont disponibles pour i386 et m68k, et la version 2.0.1 pour les alpha, vous les trouverez à

- Intel x86:
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.4.bin.i386.tar.gz`
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-crypt-2.0.4.bin.i386.tar.gz`
- Alpha:
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.1.bin.alpha-linux.tar.gz`
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-crypt-2.0.1.bin.alpha-linux.tar.gz`
- m68k:
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.4-m68k-linux.bin.tar.gz`
  - `ftp://prep.ai.mit.edu/pub/gnu/glibc-crypt-2.0.4-m68k-linux.bin.tar.gz`

Il y a des restrictions à l'exportation de l'extension crypt. Les utilisateurs non-ressortissants des États-Unis devraient l'obtenir à

`ftp://ftp.ifi.uio.no/pub/gnu` .

Si vous utilisez une distribution Red Hat, vous pouvez obtenir les RPMs pour la glibc 2 à `ftp://ftp.redhat.com/pub/redhat/` . La glibc 2 est la bibliothèque C principale de la nouvelle distribution Red Hat 5.0.

Si vous utilisez une distribution Debian, vous pouvez obtenir les paquetages pour glibc 2 à `ftp://ftp.debian.org/debian/dists/unstable/main/` . Les fichiers sont nommés libc6. Glibc 2 fait maintenant partie du paquetage de base de la version hamm de Debian, et sera la libc principale quand Debian 2.0 sortira.

4 Installation comme bibliothèque de test

Cette section couvre l'installation de glibc 2 comme bibliothèque de test. Tout ce que vous compilerez sera lié à vos bibliothèques existantes sauf si vous donnez des paramètres supplémentaires pour les lier aux nouvelles bibliothèques. Il semble que les chemins d'accès soient compilés dans un certain nombre de fichiers, et vous devrez probablement installer la bibliothèque à partir des sources.

4.1 Compilation et installation

4.1.1 Éléments requis

- À peu pres 150 Mo d'espace disque libre
- GNU make 3.75
- gcc >= 2.7.2 (ou mieux, 2.7.2.1)
- binutils 2.8.1 (pour les alpha vous devez utiliser une mise à jour temporaire ou *snapshot*)
- bash 2.0
- autoconf 2.12 (si vous changez configure.in)
- texinfo 3.11

Sur un i586 à 133 MHz avec 64 Mo de RAM, il faut environ trois heures pour compiler les bibliothèques complètes avec les extensions. Sur un i686 à 200 MHz chargé, il faut environ une demi-heure.

4.1.2 Extraction des sources

Vous devez extraire les sources des archives pour pouvoir les compiler. La meilleure façon de procéder est de faire ainsi :

```
tar xzf glibc-2.0.6.tar.gz
cd glibc-2.0.6
tar xzf ../glibc-linuxthreads-2.0.6.tar.gz
tar xzf ../glibc-crypt-2.0.6.tar.gz
tar xzf ../glibc-localedata-2.0.6.tar.gz
```

Ceci mettra les répertoires linuxthreads, crypt et localedate dans le répertoire glibc-2.0.6 où configure pourra trouver ces extensions.

4.1.3 Configuration

Dans le répertoire glibc-2.0.6, créez un répertoire appelé compile, et déplacez-vous dedans. Tout le travail doit être effectué dans ce répertoire, ce qui simplifiera le nettoyage (les développeurs ne se sont pas très occupés de rendre 'make clean' parfait pour l'instant).

```
mkdir compile
cd compile
```

Lancez `../configure`. Pour utiliser les paquetages d'extension, vous devez les spécifier avec –enable-add-ons, comme –enable-add-ons=linuxthreads,crypt,localedata. Vous devez aussi choisir un répertoire de destination pour l'installation. /usr/i486-linuxglibc2 est un bon choix. La ligne de commande de configure pour ceci serait :

```
../configure --enable-add-ons=linuxthreads,crypt,localedata --prefix=/usr/i486-linuxglibc2
```

4.1.4 Compilation et installation

Pour compiler et vérifier, lancez :

```
make
make check
```

Si le 'make check' réussit, installez la bibliothèque :

```
make install
```

4.2 Mise à jour du chargeur de liens dynamique

1. Créez un lien à partir du nouvel `ld.so` vers `/lib/ld-linux.so.2` :

   ```
   ln -s /usr/i486-linuxglibc2/lib/ld-linux.so.2 /lib/ld-linux.so.2
   ```

C'est la seule bibliothèque dont l'emplacement est fixé une fois qu'un programme est lié, et l'utilisation d'un lien dans `/lib` facilitera le passage à glibc en tant que bibliothèque C principale quand la version stable sortira.

2. Éditez `/etc/ld.so.conf`. Vous devez ajouter le chemin vers le répertoire lib dans lequel se trouvent les nouvelles bibliothèques à la fin du fichier, qui sera `<préfixe>/lib`, comme `/usr/i486-linuxglibc2/lib` dans l'exemple ci-dessus. Après avoir modifié `/etc/ld.so.conf`, lancez

   ```
   ldconfig -v
   ```

4.3 Configuration pour gcc

La dernière étape de l'installation est la mise à jour de `/usr/lib/gcc-lib` pour que gcc sache comment utiliser les nouvelles bibliothèques. D'abord vous devez dupliquer la configuration existante. Pour savoir quelle est la configuration en cours, utilisez l'option -v de gcc :

```
% gcc -v
Reading specs from /usr/lib/gcc-lib/i486-unknown-linux/2.7.2.2/specs
gcc version 2.7.2.2
```

Dans ce cas, i486-unknown-linux est le système, et 2.7.2.2 est la version. Vous devez copier `/usr/lib/gcc-lib/<système>` vers le nouveau répertoire système de test :

```
cd /usr/lib/gcc-lib/
cp -r i486-unknown-linux i486-linuxglibc2
```

Allez dans le nouveau répertoire système de test et dans le répertoire version

```
cd /usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2
```

et éditez le fichier `specs` se trouvant dans ce répertoire. Dans ce fichier, changez `/lib/ld-linux.so.1` en `/lib/ld-linux.so.2`. Vous devrez aussi enlever toutes les expressions `%{...:-lgmon}` du fichier, puisque glibc n'utilise pas la bibliothèque gmon pour les optimisations. Vous trouverez un fichier specs d'exemple dans la section 9 (Fichiers specs d'exemple).

4.4 Mise à jour des liens des fichiers d'en-tête

Vous devez créer des liens dans votre nouveau répertoire d'en-têtes vers d'autres répertoires d'en-têtes :

```
cd /usr/i486-linuxglibc2/include
ln -s /usr/src/linux/include/linux
ln -s /usr/src/linux/include/asm
ln -s /usr/X11R6/include/X11
```

Vous avez peut-être d'autres bibliothèques comme ncurses qui nécessitent d'avoir leurs fichiers d'en-têtes dans ce répertoire. Vous devriez copier ou faire un lien vers ces fichiers depuis `/usr/include`. (Certaines bibliothèques auront besoin d'être recompilées avec glibc2 pour pouvoir fonctionner avec glibc2. Dans ces cas, compilez simplement et installez les paquetages dans `/usr/i486-linuxglibc2`.)

4.5 Test de l'installation

Pour tester l'installation, créez le programme suivant dans un fichier appelé glibc.c :

```
#include <stdio.h>

main()
{
    printf("hello world!\n");
}
```

et compilez-le avec les options "-b <répertoire d'installation> -nostdinc -I<répertoire d'installation>/include -I/usr/lib/gcc-lib/<nouveau système>/<version de gcc>/include" :

```
% gcc -b i486-linuxglibc2 -nostdinc -I/usr/i486-linuxglibc2/include -I/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include glibc.c -o glibc
```

Utilisez ldd pour vérifier que le programme a été lié avec glibc2, et non avec votre ancienne libc :

```
% ldd glibc
libc.so.6 => /usr/i486-linuxglibc2/lib/libc-2.0.6.so (0x4000d000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

S'il compile, que les liens sont effectués et qu'il génère "hello world!" quand vous le lancez, l'installation a fonctionné.

5 Installation comme bibliothèque C principale

Cette section couvre l'installation de glibc2 comme bibliothèque C principale. Tous les nouveaux programmes que vous compilerez seront liés avec cette bibliothèque, sauf si vous utilisez des options de compilation spéciales pour le lier avec une autre version.

Si vous utilisez Redhat ou Debian et avez chargé les fichier rpm ou deb adéquats, voyez les instructions d'installation Redhat ou Debian. Vous pouvez alors sauter cette section.

5.1 Construction de la bibliothèque à partir des sources

Cette section explique comment compiler glibc 2 et ses extensions à partir des sources. Vous devez compiler la bibliothèque si vous voulez changer les options d'optimisation ou de configuration ou utiliser un paquetage pour lequel vous n'avez pas de binaire.

5.1.1 Éléments requis

- Environ 150 Mo d'espace disque libre
- GNU make 3.75
- gcc >= 2.7.2 (ou mieux, 2.7.2.1)
- binutils 2.8.1 (pour les alpha vous aurez besoin d'une version 'snapshot')

- bash 2.0
- autoconf 2.12 (si vous changez configure.in)
- texinfo 3.11

Sur un i586 à 133 MHz avec 64 Mo de RAM, il faut environ trois heures pour compiler toutes les bibliothèques avec les extensions. Sur un i686 à 200 MHz chargé, il faut environ une demi-heure.

5.1.2 Extraction des sources

Vous devez extraire les sources des archives pour compiler. La meilleure façon de le faire est ainsi :

```
tar xzf glibc-2.0.6.tar.gz
cd glibc-2.0.6
tar xzf ../glibc-linuxthreads-2.0.6.tar.gz
tar xzf ../glibc-crypt-2.0.6.tar.gz
tar xzf ../glibc-localedata-2.0.6.tar.gz
```

Ceci mettra les répertoires linuxthreads, crypt et localedata dans le répertoire glibc-2.0.6 où configure pourra trouver ces extensions.

5.1.3 Configuration

Dans le répertoire `glibc-2.0.6`, créez un répertoire nommé compile, et allez dedans. Tout le travail sera fait dans ce répertoire, ce qui simplifiera le nettoyage (les développeurs ne se sont pas vraiment soucié du 'make clean' pour l'instant).

```
mkdir compile
cd compile
```

Lancez `../configure`. Pour utiliser les paquetages supplémentaires, vous devez les spécifier avec –enable-add-ons, comme –enable-add-ons=linuxthreads,crypt,localedata. Vous devrez aussi sûrement spécifier les chemins où elle sera installée. Pour coller aux distributions Linux normales, spécifiez –prefix=/usr. (Quand on spécifie le préfixe `/usr` sur un système Linux, configure sait ajuster les autres chemins pour placer libc.so et d'autres bibliothèques importantes dans `/lib`.) La ligne de commande complète de configure serait :

```
../configure --enable-add-ons=linuxthreads,crypt,localedata --prefix=/usr
```

5.1.4 Compilation

Pour compiler et vérifier, lancez :

```
make
make check
```

5.2 Préparation à l'installation

Vous devez maintenant déplacer certains fichiers pour vous préparer à l'arrivée de la nouvelle bibliothèque, que vous l'installiez à partir des sources ou de binaires. Tous les nouveaux programmes compilés seront liés à glibc, mais les vieux programmes qui ne sont pas liés en statique dépendront encore de libc 5, et vous ne pouvez donc pas écraser l'ancienne version.

1. Créez un nouveau répertoire pour y mettre les anciens fichiers :

   ```
   mkdir -p /usr/i486-linuxlibc5/lib
   ```

2. Les vieux fichiers d'en-tête doivent être évacués de `/usr/include` :

   ```
   mv /usr/include /usr/i486-linuxlibc5/include
   ```

3. Créez un nouveau répertoire d'en-têtes et activez les liens vers d'autres en-têtes :

   ```
   mkdir /usr/include

   ln -s /usr/src/linux/include/linux /usr/include/linux
   ln -s /usr/src/linux/include/asm /usr/include/asm
   ln -s /usr/X11R6/include/X11 /usr/include/X11
   ln -s /usr/lib/g++-include /usr/include/g++
   ```

   Les liens devront être ajustés au besoin selon votre distribution. Rien que la Slackware installe les en-têtes g++ dans `/usr/local/g++-include`, alors que Debian met les en-têtes dans `/usr/include/g++`, et fait un lien de `/usr/lib/g++-include` vers `/usr/include/g++`. Dans d'autres cas, vous voudrez sûrement déplacer le répertoire d'en-têtes g++ d'origine à son ancien emplacement `/usr/include`.

4. Remettre des fichiers d'en-têtes et des liens supplémentaires. Certaines bibliothèques non standards comme ncurses installent des fichiers dans `/usr/include` ou installent un lien vers leur répertoire d'en-tête dans `/usr/include`. Ces fichiers et liens doivent être remis en place pour pouvoir utiliser les bibliothèques supplémentaires correctement.

5. Ajoutez le nouveau répertoire de bibliothèque (comme `/usr/i486-linuxlibc5/lib`) *en haut* de votre fichier `/etc/ld.so.conf`. Vous devriez avoir ld.so-1.8.8 ou mieux installé pour éviter d'avoir des messages bizarres une fois que glibc sera installée.

6. Déplacez ou copiez toutes les anciennes bibliothèques C dans le nouveau répertoire.

   ```
   mv /usr/lib/libbsd.a /usr/i486-linuxlibc5/lib
   mv /usr/lib/libc.a /usr/i486-linuxlibc5/lib
   mv /usr/lib/libgmon.a /usr/i486-linuxlibc5/lib
   mv /usr/lib/libm.a /usr/i486-linuxlibc5/lib
   mv /usr/lib/libmcheck.a /usr/i486-linuxlibc5/lib
   mv /usr/lib/libc.so /usr/i486-linuxlibc5/lib
   mv /usr/lib/libm.so /usr/i486-linuxlibc5/lib
   cp /lib/libm.so.5.* /usr/i486-linuxlibc5/lib
   cp /lib/libc.so.5.* /usr/i486-linuxlibc5/lib
   ```

   `libm.so.5` et `libc.so.5` doivent être copiées et non déplacées si `/usr` est une partition différente de `/`, parce qu'elles sont nécessaires aux programmes utilisés pour démarrer Linux et doivent être situées sur la partition racine.

7. Déplacez les fichiers `/usr/lib/*.o` dans le nouveau répertoire.

```
mv /usr/lib/crt1.o /usr/i486-linuxlibc5/lib
mv /usr/lib/crti.o /usr/i486-linuxlibc5/lib
mv /usr/lib/crtn.o /usr/i486-linuxlibc5/lib
mv /usr/lib/gcrt1.o /usr/i486-linuxlibc5/lib
```

8. Mettez à jour votre cache de bibliothèque après avoir déplacé vos bibliothèques.

```
ldconfig -v
```

5.3 Installation à partir du paquetage binaire

Si vous installez glibc à partir de binaires précompilés, vous devez :

```
cd /
gzip -dc glibc-2.0.bin.i386.tar.gz | tar tvvf -
gzip -dc glibc-crypt-2.0.bin.i386.tar.gz | tar tvvf -
ldconfig -v
```

Si vous avez une architecure ou une version différente, substituez les noms de fichiers adéquats.

5.4 Installation à partir des sources

Pour installer la bibliothèque à partir des sources, lancez :

```
make install
ldconfig -v
```

5.5 Mise à jour des specs gcc

L'étape finale de l'installation (à la fois pour les installations binaires et sources) est de mettre à jour le fichier `specs` de gcc pour que vous puissiez lier vos programmes proprement. Pour déterminer quel fichier specs est utilisé par gcc, lancez :

```
% gcc -v
reading specs from /usr/lib/gcc-lib/i486-unknown-linux/2.7.2.2/specs
gcc version 2.7.2.2
```

Dans ce cas, i486-unknown-linux est le système, et 2.7.2.2 est la version. Vous devez copier `/usr/lib/gcc-lib/`<`système`> dans l'ancien répertoire système :

```
cd /usr/lib/gcc-lib/
cp -r i486-unknown-linux i486-linuxlibc5
```

Allez dans le répertoire d'origine et dans le répertoire de version

```
cd /usr/lib/gcc-lib/i486-unknown-linux/2.7.2.2
```

et éditez le fichier `specs` que vous y trouverez. Dans ce fichier, changez `/lib/ld-linux.so.1` en `/lib/ld-linux.so.2`. Vous devrez aussi enlever toutes les expressions `%{...:-lgmon}` de ce fichier, puisque glibc n'utilise pas la bibliothèque gmon pour les optimisations. Un fichier specs d'exemple se trouve dans la section 9 (fichiers specs d'exemple).

5.6 Test de l'installation

Pour tester l'installation, créez le programme suivant dans un fichier appelé glibc.c :

```
#include <stdio.h>

main()
{
    printf("hello world!\n");
}
```

et compilez le programme.

```
% gcc glibc.c -o glibc
```

Utilisez ldd pour vérifier que le programme a été lié avec glibc2, et non avec votre ancienne libc :

```
% ldd glibc
libc.so.6 => /lib/libc.so.6 (0x4000e000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Si ça compile et que ça génère "hello world!" quand vous le lancez, l'installation est réussie.

6 Compilation avec la libc non principale

Il y a des fois où vous voudrez utiliser une bibliothèque différente pour compiler vos programmes. Cette section explique comment faire, en utilisant les noms de répertoires et d'installation utilisés dans les exemples des deux sections précédentes. Souvenez-vous de changer les noms pour coller à votre configuration.

6.1 Avertissement concernant l'utilisation de libcs non principales

Avant de compiler un programme utilisé dans le processus de démarrage, rappelez-vous que si le programme est lié dynamiquement, et est utilisé avant que les partitions non-racines soient montées, toutes les bibliothèques liées doivent être sur la partition racine. En suivant la procédure d'installation de la section précédente pour installer glibc comme bibliothèque C principale, la vieille libc reste dans `/lib`, qui sera sur votre partition racine. Ceci veut dire que tous vos programmes fonctionneront encore lors du démarrage. Cependant, si `/usr` est sur une partition différente et que vous installez glibc comme bibliothèque de test dans `/usr/i486-linuxglibc2`, tous les nouveaux programmes que vous compilerez avec glibc ne fonctionneront pas tant que la partition `/usr` ne sera pas montée.

6.2 Compilation des programmes avec une glibc de test

Pour compiler un programme avec glibc en installation de test, vous devez mettre à zéro les chemins d'en-tête pour qu'ils pointent vers les en-têtes glibc. En spécifiant "-nostdinc", vous annulerez les chemins normaux, et "-I/usr/i486-linuxglibc2/include" pointera vers les entêtes de glibc. Vous devrez aussi spécifier les en-têtes gcc, que l'on trouve dans `/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include` (en supposant que vous ayez installé la bibliothèque de test dans i486-linuxglibc2 avec gcc version 2.7.2.2).

Pour lier un programme à une glibc de test, vous devez spécifier la configuration gcc. Vous le faites avec l'option "-b i486-linuxglibc2".

Pour la plupart des programmes, vous pouvez spécifier ces nouvelles options en les ajoutant aux options de makefile `$CFLAGS` et `$LDFLAGS` :

```
CFLAGS = -nostdinc -I/usr/i486-linuxglibc2/include -I/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include -b i486-linuxglibc2
LDFLAGS = -b i486-linuxglibc2
```

Si vous utilisez un script configure, définissez les variables shell `$CFLAGS` et `$LDFLAGS` (en utilisant env/setenv pour csh/tcsh ou set/export pour sh/bash/etc) avant de lancer configure. Les makefiles générés par celui-ci devraient contenir les variables `$CFLAGS` et `$LDFLAGS` correctes. Tous les scripts configure ne tiendront pas compte des variables, et par conséquent vous devriez vérifier après avoir lancé configure et éditer les makefiles à la main si nécessaire.

Si les programmes que vous compilez n'appellent que gcc (et pas cpp ou les binutils directement), vous pouvez utiliser le script suivant pour économiser la spécification des options à chaque fois :

```
#!/bin/bash
/usr/bin/gcc -b i486-linuxglibc2 -nostdinc \
             -I/usr/i486-linuxglibc2/include \
             -I/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include "$@"
```

Vous pourrez alors utiliser ce script à la place de "gcc" pour compiler.

6.3 Compilation des programmes avec libc 5 quand glibc est la bibliothèque principale

Pour compiler un programme avec vos anciennes bibliothèques quand vous avez installé glibc comme bibliothèque principale, vous devez mettre à zéro les chemins d'en-têtes vers les anciennes en-têtes. En spécifiant "-nostdinc", vous annulerez les chemins normaux et "-I/usr/i486-linuxlibc5/include" pointera vers les en-têtes libc 5. Vous devez aussi indiquer "-I/usr/lib/gcc-lib/i486-linuxlibc5/2.7.2.2/include" pour inclure les en-têtes spécifiques à gcc. Rappelez-vous d'ajuster ces chemins selon la manière dont vous avez nommé les nouveaux répertoires et selon votre version de gcc.

Pour lier un programme à votre ancienne libc, vous devez spécifier la configuration de gcc. Vous le faites en utilisant l'option "-b i486-linuxlibc5".

Pour la plupart des programmes, vous pouvez indiquer ces nouvelles options en les ajoutant aux options de makefile `$CFLAGS` et `$LDFLAGS` :

```
CFLAGS = -nostdinc -I/usr/i486-linuxlibc5/include -I/usr/lib/gcc-lib/i486-linuxlibc5/2.7.2.2/include -b i486-linuxlibc5
LDFLAGS = -b i486-linuxlibc5
```

Si vous utilisez un script configure, définissez les variables shell `$CFLAGS` et `$LDFLAGS` (en utilisant env/setenv pour csh/tcsh, ou set/export pour sh/bash/etc) avant de lancer configure. Les makefiles générés par celui-ci devraient contenir les variables `$CFLAGS` et `$LDFLAGS` corrects. Tous les scripts configure ne verront pas les variables, et vous devriez donc vérifier après avoir lancé configure et éditer les makefiles si nécessaire.

Si les programmes que vous compilez n'appellent que gcc (et pas cpp ou binutils directement), vous pouvez utiliser le script suivant pour éviter de devoir indiquer toutes les options à chaque fois :

```
#!/bin/bash
/usr/bin/gcc -b i486-linuxlibc5 -nostdinc \
             -I/usr/i486-linuxlibc5/include \
             -I/usr/lib/gcc-lib/i486-linuxlibc5/2.7.2.2/include "$@"
```

Vous pouvez alors utiliser ce script à la place de "gcc" pour la compilation.

7 Compilation des programmes C++

Libg++ utilise des parties de la bibliothèque mathématique, elle est donc liée à libm. Puisque votre bibliothèque libg++ existante aura été compilée avec votre ancienne bibliothèque, vous devrez recompiler libg++ avec glibc ou obtenir une copie binaire. Les dernières sources de libg++, en même temps qu'un binaire lié à glibc (pour x86) se trouvent à `ftp://ftp.yggdrasil.com/private/hjl/` .

7.1 Installation de libg++ pour une installation glibc de test

Si vous avez installé glibc comme bibliothèque de test, vous devez installer les fichiers dans le répertoire dans lequel vous avez installé glibc (comme `/usr/i486-linuxglibc2` pour l'exemple des sections précédentes). Si vous installez à partir du paquetage binaire (ce que je recommanderais, puisque je n'ai jamais eu de chance pour compiler libg++ de cette manière), vous devez extraire les fichiers dans un répertoire temporaire et déplacer tous les fichiers `usr/lib/` dans le répertoire `<répertoire install>/lib/`, les fichiers `usr/include/` dans le répertoire `<répertoire install>/include/` (rappelez-vous d'effacer le lien `include/g++` avant !), et les fichiers `usr/bin/` dans le répertoire `<répertoire install>/bin/`.

7.2 Installation de libg++ pour une installation glibc principale

Si vous avez installé glibc comme bibliothèque principale, vous devez d'abord déplacer vos anciens fichiers libg++ dans l'ancien répertoire libc si vous voulez encore pouvoir compiler des programmes g++ avec votre ancienne libc. La meilleure façon de procéder est probablement d'installer une nouvelle copie de libg++ compilée avec libc 5 comme dans la section précédente, et ensuite d'installer la version glibc normalement.

7.3 Compilation de programmes C++ avec la libc non principale

Si vous essayez de compiler un programme C++ avec une libc non principale, vous devrez inclure le répertoire d'en-têtes g++, qui dans les exemples ci-dessus serait `/usr/i486-linuxglibc2/include/g++` pour une installation glibc de test ou `/usr/i486-linuxlibc5/include/g++` pour une installation glibc principale. On peut faire cela en général en ajoutant à la variable `$CXXFLAGS` :

```
CXXFLAGS = -nostdinc -I/usr/i486-linuxglibc2/include -I/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include -I/usr/i486-linuxlibc5/include/g++ -b i486-linuxglibc2
```

8 Indiquer les bogues

Si vous pensez que la bibliothèque est vérolée, veuillez d'abord lire la FAQ. Il se peut que d'autres personnes aient eu ce problème et qu'il y ait une solution simple. Vous devriez aussi regarder la partie "Outils recommandés pour l'installation de la bibliothèque C de GNU" dans le fichier `INSTALL` puisque certains bogues proviennent des outils et non de glibc.

Une fois que vous avez trouvé un bug, assurez-vous que c'en est vraiment un. Une bonne manière de le faire est de regarder si la bibliothèque C GNU se comporte de la même manière qu'une autre bibliothèque C. S'il en est ainsi, vous vous êtes trompé et les bibliothèques ont raison (mais pas forcément). Dans le cas contraire, l'une des bibliothèques a probablement tort.

Ensuite, dirigez-vous vers `http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl` , et parcourez la base de données des bugs. Vérifiez bien que le problème n'a pas déjà été indiqué. Vous devriez aussi jeter un coup d'oeil au fichier `BUGS` (ditribué avec glibc) pour prendre connaissance des bugs connus.

Une fois que vous êtes sûr d'avoir trouvé un bug, essayez de le réduire au plus petit test pratique qui reproduit le problème. Dans le cas d'une bibliothèque C, vous ne devrez probablement en être réduit qu'à un appel de fonction de la bibliothèque, si possible. Ceci ne devrait pas être trop difficile.

L'étape finale une fois que vous avez un exemple simple de test est d'indiquer le bug. En indiquant un bug, envoyez votre exemple de test, les résultats que vous avez obtenus, ce que vous pensez être le problème (si vous avez pensé à quelque chose), le type de votre système, les versions de la bibliothèque C GNU, du compilateur GNU CC et des GNU binutils que vous utilisez. Ajoutez aussi les fichiers `config.status` et `config.mak` créés en lançant `configure` ; ils seront dans le répertoire qui était le répertoire courant quand vous avez lancé `configure`.

Vous devez envoyer tous les rapports de bug pour la bibliothèque C GNU en utilisant le script shell `glibcbug` livré avec la libc GNU à

`bugs@gnu.org` (l'ancienne adresse `bugs@gnu.ai.mit.edu` fonctionne encore), ou par l'intermédiaire de l'interface Web de GNATS à `http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl` .

Les suggestions et les questions doivent être envoyées à la liste de distribution à `bugs-glibc@prep.ai.mit.edu` . Si vous ne lisez pas le groupe de gnews gnu.bug.glibc, vous pouvez vous abonner à la liste en demandant à `bug-glibc-request@prep.ai.mit.edu` .

Veuillez s'il vous plait ne pas envoyer de rapport de bug concernant la bibliothèque C GNU à <bug-gcc@prep.ai.mit.edu>. Cette liste s'occupe des rapports de bug de GNU CC. GNU CC et la bibliothèque C GNU sont des entités distinctes maintenues par des personnes différentes.

9 Fichiers specs d'exemple

Voici ci-inclus un fichier d'exemple `specs` pour glibc 2 que gcc utilise pour la compilation et la liaison dynamique. On devrait le trouver dans le répertoire `/usr/lib/gcc-lib/<nouveau répertoire>/<version gcc>`. Si vous utilisez un système x86, vous pouvez copier exactement cette section dans le fichier.

```
*asm:
%{V} %{v:%{!V:-V}} %{Qy:} %{!Qn:-Qy} %{n} %{T} %{Ym,*} %{Yd,*} %{Wa,*:%*}

*asm_final:
%{pipe:-}

*cpp:
%{fPIC:-D__PIC__ -D__pic__} %{fpic:-D__PIC__ -D__pic__} %{!m386:-D__i486__} %{posix:-D_POSIX_SOURCE} %{pthread:-D_REENTRANT}
```

```
*cc1:
%{profile:-p}

*cc1plus:


*endfile:
%{!shared:crtend.o%s} %{shared:crtendS.o%s} crtn.o%s

*link:
-m elf_i386 %{shared:-shared}   %{!shared:     %{!ibcs:       %{!static:       %{rdynamic:-export-dynamic}     %{!dynamic-linker:-dynamic-linker /lib/ld-linux.so.2}}  %{static:-static}}}

*lib:
%{!shared: %{pthread:-lpthread}        %{profile:-lc_p} %{!profile: -lc}}

*libgcc:
-lgcc

*startfile:
%{!shared:      %{pg:gcrt1.o%s} %{!pg:%{p:gcrt1.o%s}                  %{!p:%{profile:gcrt1.o%s}                         %{!profile:crt1.o%s}}}}    crti.o%s %{!shared:crtbegin.o%s} %{shared:crtbeginS.o%s}

*switches_need_spaces:


*signed_char:
%{funsigned-char:-D__CHAR_UNSIGNED__}

*predefines:
-D__ELF__ -Dunix -Di386 -Dlinux -Asystem(unix) -Asystem(posix) -Acpu(i386) -Amachine(i386)

*cross_compile:
0

*multilib:
. ;
```

10 Divers

10.1 Informations supplémentaires

10.1.1 Pages Web

- *Page d'accueil de la bibliothèque C GNU chez FSF* `http://www.gnu.org/software/libc/libc.html`
- *Utilisation de GNU Libc 2 avec Linux* `http://www.imaxx.net/~thrytis/glibc/`
- *Installer glibc-2 sur Linux* `http://huizen.dds.nl/~frodol/glibc/`
- *HOWTO libc5 vers libc6 Debian* `http://www.gate.net/~storm/FAQ/libc5-libc6-Mini-HOWTO.html`

10.1.2 Groupes de news

- *comp.os.linux.development.system*
- *comp.os.linux.development.apps*
- *linux.dev.kernel*
- *gnu.bugs.glibc*

10.1.3 Listes de distribution

Liste de discussion Glibc 2 Linux.

Cette liste est destinée à la discussion entre les utilisateurs Linux qui ont installé glibc2, la nouvelle bibliothèque C. Les sujets peuvent comprendre des problèmes de compatibilité et des questions sur la compilation de code dans un environnement Linux/glibc. Pour s'abonner, envoyer un courrier à *Majordomo@ricardo.ecn.wfu.edu* `mailto:Majordomo@ricardo.ecn.wfu.edu` avec dans le corps "subscribe glibc-linux <votre adresse email>".

10.2 Remerciements

J'ai volé une bonne partie de ces informations sur la *page web de GNU Libc* `http://www.gnu.org/software/libc/libc.html` et de l'annonce de glibc 2 par Ulrich Drepper <drepper@gnu.ai.mit.edu> et ses commentaires. Andreas Jaeger <aj@arthur.rhein-neckar.de> a alimenté une partie de la section sur l'indication des bugs.

Les personnes suivantes ont fourni des informations et un retour sur ce document :

- Allex <allex@ms2.accmail.com.tw>
- Mark Brown <M.A.Brown-4@sms.ed.ac.uk>
- Ulrich Drepper <drepper@gnu.ai.mit.edu>
- Scott K. Ellis <ellis@valueweb.net>
- Aron Griffis <agriffis@coat.com>
- Andreas Jaeger <aj@arthur.rhein-neckar.de>
- Frodo Looijaard <frodol@dds.nl>
- Ryan McGuire <rmcguire@freenet.columbus.oh.us>
- Shaya Potter <spotter@capaccess.org>
- Les Schaffer <godzilla@futuris.net>
- Andy Sewell <puck@pookhill.demon.co.uk>
- Gary Shea <shea@gtsdesign.com>
- Stephane <sr@adb.fr>
- Jan Vandenbos <jan@imaxx.net>

Les traductions de ce document sont faites par :

- chinois : Allex <allex@ms2.accmail.com.tw>
- français : Géraud Canet<canet@labri.u-bordeaux.fr>
- japonais : Kazuyuki Okamoto <ikko-@pacific.rim.or.jp>

10.3 Retour d'informations

En plus d'écrire ce HOWTO, maintenir la page *glibc 2 pour Linux* `http://www.imaxx.net/~thrytis/glibc` , et l'utiliser sur ma machine, je n'ai rien à voir avec le projet glibc. Je suis loin d'être un expert dans ce domaine, bien que j'essaie de résoudre les problèmes qu'on m'envoie par courrier électronique. J'apprécie tout retour, correction ou suggestion que vous pourriez me faire. Veuillez les envoyer à *ejg3@cornell.edu* `mailto:ejg3@cornell.edu` .

10.4 Copyright