Glibc 2 HOWTO

Eric Green, ejg3@cornell.edu. Översättare: Linus Åkerlund, uxm165t@tninet.se. v1.5, 8 February 1998.
Översättning: 26 juli 1998.

Glibc2-HOWTOn tar upp installering och användning av GNU C-bibliotek version 2 (libc 6) på Linux-system.

Innehåll

1 Inledning

1.1 Om glibc 2.

Glibc 2 är den senaste versionen av GNU C-biblioteket. För tillfället kan det köras oförändrat på GNU/Hurd-system och Linux i386-, m68k- och Alpha-system. Stöd för Linux PowerPC, MIPS, Sparc, Sparc 64 och Arm kommer finnas i och med version 2.1. I framtiden kommer stöd för andra arkitekturer och operativsystem läggas till.

Under Linux används glibc 2 som libc med versionsnummer 6, alltså som uppföljaren till Linux libc 5. Det avses av Linux libc-utvecklarna ersätta libc 5 så småningom. I och med 2.0.6 anses glibc vara stabilt. Version 2.1 (som bör vara ute snart) kommer vara klar att användas i alla sammanhang, och kommer stödja flera arkitekturer och ha nya funktioner.

Tre valfria tillägg finns tillgängliga för glibc 2:

Crypt

UFC-crypt-paketet. Det hålls separat på grund av export-restriktioner.

LinuxThreads

En implementering av Posix 1003.1c "pthread"-gränssnittet.

Locale data

Innehåller alls som behövs för att skapa locale-datafiler, för att utnyttja glibcs internationaliserings-funktioner.

Crypt- och LinuxThreads-tilläggen är starkt rekommenderade... Om du inte använder dem riskerar du att ditt glibc inte är kompatibelt med biblioteken på andra system. (Om du inte vill använda dem måste du lägga till parametern –disable-sanity-checks när du kör configure.)

1.2 Om detta dokument.

Denna HOWTO tar upp installering av glibc 2-biblioteket på ett existerande Linux-system. De är riktat till användare av Intel-baserade system som för tillfället använder libc 5, men användare av andra system och alternativa bibliotek (som t.ex. glibc 1) bör också kunna använda denna information genom att sätta in passande filnamn och arkitekturer där detta ska göras.

Den senaste versionen av denna HOWTO hittar du hos *Linux Documentation Project* `<http://sunsite.unc.edu/LDP>` och `<http://www.imaxx.net/~thrytis/glibc/Glibc2-HOWTO.html>`.

Den svenska versionen ska finnas tillgänglig från *www.swe-doc.linux.nu* `<http://www.swe-doc.linux.nu/>` och *min hemsida* `<http://user.tninet.se/~uxm165t/linux_doc.html>`.

1.3 Nyligen gjorda ändringar i detta dokument.

Skillnader mellan version 1.5 och 1.4:

- Indexering tillagt av Ed Bailey.
- Ändrade min epost-adress (original-författarens. övers.anm.)

Skillnader mellan version 1.4 och 1.3:

- Ändrade status från experimentellt till stabilt.
- Uppdaterade listan över utvecklingsversioner.
- Uppdaterade senaste versionen till 2.0.6.

2 Att välja installeringsmetod.

Det finns några olika sätt att installera glibc. Du kan installera biblioteken som ett test, och använda de existerande biblioteken som standard, vilket låter dig pröva de nya biblioteken genom att använda olika parametrar då du kompilerar dina program. Installerar du på detta sätt är det enkelt att ta bort glibc i framtiden (men detta gör att inget program som är länkat med glibc fungerar längre, då biblioteken är borttagna). Vill du använda glibc som testbibliotek så måste du kompilera det från källkod. Det finns ingen binärdistribution för att installera biblioteken på detta sätt. Denna installering beskriv i 4 (Installera som ett test-bibliotek).

Det andra sättet att installera, som beskrivs i det här dokumentet, är att använda glibc som ditt primära bibliotek. alla nya program som du kompilerar på ditt system kommer använda glibc, men du kan länka programmen med ditt gamla bibliotek genom att ange andra parametrar då du kompilerar. Du kan antingen installera biblioteken från binärer eller kompilera själv. Om du vill ändra optimerings- eller konfigurerings-alternativ eller använda ett tillägg som inte distribueras med binärpaketet så måste du skaffa källkoden och kompilera den. Denna installeringsprocedur beskrivs i 5 (Installera som primärt C-bibliotek).

Frodo Looijaard beskriver ännu ett annat sätt att installera glibc. Hans metod går ut på att installera glibc som sekundärt bibliotek och konfigurera en korskompilator (cross compiler) för att kompilera med glibc. Installeringsproceduren för denna metod är mer komplicerad än test-biblioteksinstalleringen som beskrivs i detta dokument, men gör det mycket enklare att kompilera när du länkar med glibc. Denna metod beskrivs i hans dokument *Installing glibc-2 on Linux* `<http://huizen.dds.nl/~frodol/glibc/>`.

Om du kör Debian 1.3 men inte vill uppgradera till en instabil version av Debian för att kunna använda glibc så beskriver *Debian libc5 to libc6 Mini-HOWTO* `<http://www.gate.net/~storm/FAQ/libc5-libc6-Mini-HOWTO.html>` hur du kan uppgradera ditt system med Debian-paket.

Om du installerar glibc 2 på ett viktigt system så bör du nog välja test-installeringen. Även om det inte finns några buggar så kommer vissa program kräva vissa modifikationer innan de kan kompileras, p.g.a. ändringar i funktions-prototyper och typer.

3 Skaffa biblioteket.

glibc 2 består av glibc-paketet och tre valfria tilläggspaket; LinuxThreads, Locale och Crypt. Källkoden hittar du bl.a. på:

- `<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>`

Den tar upp ungefär 150 MB diskutrymme för en fullständig kompilering och installering. Den grundläggande binärinstalleringen av bara kärnan i biblioteket är på ungefär 50 MB.

Binärpaket för 2.0.6 är inte tillgängliga. Binärpaket av version 2.0.4 finns tillgängliga för i386 och m68k, och version 2.0.1 för Alpha. Dessa hittar du på:

- 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>`

Crypt-tillägget är belagt med exportrestriktioner. Bor du utanför U.S.A. kan du hämta detta från `<ftp://ftp.ifi.uio.no/pub/gnu>`.

Om du kör en Red Hat-distribution så kan du skaffa glibc 2 som RPM-paket från `<ftp://ftp.redhat.com/pub/redhat/>`. glibc 2 är det primära C-biblioteket för Red Hat 5.0.

Om du kör en Debian-distribution kan du hämta glibc 2-paket från `<ftp://ftp.debian.org/debian/dists/unstable/main/>`. Filerna heter libc6. glibc 2 är nu med i baspaketet av hamm-versionen av Debian och kommer vara det primära C-biblioteket när Debian 2.0 kommer ut.

4 Installera som ett test-bibliotek.

Detta avsnitt tar upp installeringen av glibc 2 som ett test-bibliotek. Allt du kompilerar kommer länkas med dina existerande bibliotek om du inte ger några extraparametrar för att länka med de nya biblioteken. Det verkar som att sökvägarna kompileras in i en hel del filer, så du gör antagligen bäst i att installera biblioteket från källkod.

4.1 Kompilera och installera.

4.1.1 Förutsättningar.

- Runt 150 MB fritt diskutrymme
- GNU make 3.75
- gcc >= 2.7.2 (ännu hellre 2.7.2.1)
- binutils 2.8.1 (till Alpha behöver du ett snapshot)
- bash 2.0
- autoconf 2.12 (om du ändrar configure.in)
- texinfo 3.11

På en i586@133 med 64 MB RAM tar det runt 3 timmar att kompilera med fullständiga bibliotek och tillägg. På en laddad i686@200 tar det runt en halvtimme.

4.1.2 Packa upp källkoden.

Du måste packa upp källkoden från arkivet så att du kan kompilera den. Det bästa sättet att göra detta på är:

```
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
```

Detta kommer att placera linuxthreads-, crypt- och localedata-katalogerna i glibc-2.0.6-katalogen där configure kan hitta dessa tillägg.

4.1.3 Konfigurering.

I glibc-2.0.6-katalogen ska du skapa en katalog som heter compile och cd-a till den. Allt arbete kommer göras i denna katalog, vilket förenklar städningen. (Utvecklarna har inte brytt sig speciellt mycket om att göra 'make clean' perfekt än.)

```
mkdir compile
cd compile
```

Kör `../configure`. För att använda tilläggspaketen måste du ange dessa med –enable-add-ons, t.ex. –enable-add-ons=linuxthreads,crypt,localedata. Du måste även välja en katalog att installera i. /usr/i486-linuxglibc2 är ett bra val. configure-raden för detta blir:

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

4.1.4 Kompilera och installera.

För att kompilera och verifiera, kör:

```
make
make check
```

Om 'make check' går bra så kan du installera biblioteket:

```
make install
```

4.2 Uppdatera den dynamiska laddaren.

1. Skapa en länk från den nya `ls.so` till `/lib/ld-linux.so.2`:

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

   Detta är det enda biblioteket vars position är fast då ett program länkas, och att använda en länk i `/lib` gör det lättare att uppgradera till glibc som ditt primära C-bibliotek när den stabila versionen ges ut.

2. Editera `/etc/ld.so.conf`. Du måste, i slutet av filen, lägga till sökvägen till lib-katalogen där det nya biblioteket finns, vilket blir `<prefix>/lib`, alltså `/usr/i486-linuxglibc2/lib` i exemplet ovan. Då du modifierat `/etc/ld.so.conf` kan du köra

   ```
   ldconfig -v
   ```

4.3 Konfigurering för gcc.

Det sista steget i installeringen är att uppdatera `/usr/lib/gcc-lib`, så att gcc vet hur det ska använda de nya biblioteken. Först måste du ta en kopia av de existerande inställningarna. För att ta redan på vilken konfigurering du har för tillfället kan du använda parametern -v till gcc:

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

I detta fall är i486-unknown-linux systemet och 2.7.2.2 är versionen. Du måste kopiera `/usr/lib/gcc-lib/<system>` till den nya test-systemkatalogen:

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

Gå till din nya test-systemkatalog och versionskatalog:

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

och editera filen `specs`. I denna fil ska du byta ut `/lib/ld-linux.so.1` mot `/lib/ld-linux.so.2`. Du måste också radera alla uttryck `%{...:-lgmon}` i filen, eftersom glibc inte använder gmon-biblioteket för "profiling". Ett exempel på en specs-fil kan du hitta i avsnittet 9 (Exempel på specs-fil).

4.4 Uppdatera länkar till header-filer

Du måste skapa länkar i din nya include-katalog till andra include-kataloger:

```
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
```

Det kan också hända att du har andra bibliotek, såsom ncurses, som behöver ha sina header-filer i denna katalog. Du bör kopiera eller länka filerna från `/usr/include`. (Vissa bibliotek kan behöva kompileras om med glibc2 för att fungera med det. I dessa fall är det bara att kompilera och installera paketen i `/usr/i486-linuxglibc2`.)

4.5 Testa din installering.

För att testa installeringen, skapa följande program i filen glibc.c:

```
#include <stdio.h>

main()
{
    printf("hej världen!\n");
}
```

och kompilera det med parametrarna "-b <bas-installerings-katalog> -nostdinc -I<installerings-katalog>/include -I/usr/lib/gcc-lib/<ny system-katalog>/<gcc version>/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
```

Använd ldd för att verifiera att programmet länkades med glibc2 och inte med ditt gamla 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)
```

Om det kompilerar, länkarna stämmer och det visar "hej världen!" när det körs så har installeringen lyckats.

5 Installera som primärt C-bibliotek.

Detta avsnitt tar upp installeringen av glibc 2 som ditt primära C-bibliotek. Alla nya program du kompilerar kommer länkas med detta bibliotek om du inte anger speciella kompileringsparametrar, för att länka med en annan version.

Om du använder RedHat eller Debian och har laddat ner de rätta rpm- eller deb-filerna så kan du titta på deras installeringsinstruktioner. Du kan hoppa över det här avsnittet.

5.1 Bygga biblioteket från källkod.

Detta avsnitt förklarar hur du kompilerar källkoden till glibc 2 och tilläggen. Du måste kompilera biblioteket om du vill ändra optimeringar eller inställningar eller använda ett paket som du inte har binärt.

5.1.1 Förutsättningar.

- Runt 150 MB fritt hårddiskutrymme
- GNU make 3.75
- gcc >= 2.7.2 (ännu hellre 2.7.2.1)
- binutils 2.8.1 (till Alpha behöver du ett snapshot)
- bash 2.0
- autoconf 2.12 (om du ändrar configure.in)
- texinfo 3.11

På en i586@133 med 64 MB RAM tar det runt 3 timmar att kompilera med fullständiga bibliotek och tillägg. På en laddad i686@200 tar det runt en halvtimme.

5.1.2 Packa upp källkoden.

Du måste packa upp källkoden från arkiven så att du kan kompilera den. Det bästa sättet att göra detta på är:

```
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
```

Detta placerar linuxthreads-, crypt- och localedata-kataloger i katalogen glibc-2.0.6, där configure kan hitta dessa tillägg.

5.1.3 Konfigurering.

Skapa en katalog som heter compile i `glibc-2.0.6`-katalogen och cd-a till den. Allt arbete kommer utföras i denna katalog, vilket kommer underlätta städningen efteråt. (Utvecklarna har inte brytt sig så mycket om att få 'make clean' perfekt än.)

```
mkdir compile
cd compile
```

Kör `../configure`. För att använda tilläggspaketen måste du ange dem med –enable-add-ons, alltså t.ex. –enable-add-ons=linuxthreads,crypt,localedata. Du bör antagligen också ange var du vill installera det. För att matcha de standardiserade Linux-distributionerna kan du ange –prefix=/usr. (När prefixet anges som `/usr` vet configure om att den ska ändra sina sökvägar för att placera libc.so och andra viktiga bibliotek i `/lib`.) Hela configure-raden blir alltså:

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

5.1.4 Kompilera.

För att kompilera och verifiera, kör:

```
make
make check
```

5.2 Förberedelser inför installeringen.

Nu måste du flytta runt en del filer för att göra plats för det nya biblioteket, vare sig du installerar från källkod eller binärer. Alla nya program du kompilerar kommer länkas med glibc, men äldre program som inte är statiskt länkade kommer fortfarande vara beroende av libc 5, så du kan inte bara skriva över den gamla versionen.

1. Skapa en ny katalog att stoppa de gamla filerna i:

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

2. De gamla header-filerna måste evakueras från `/usr/include`:

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

3. Skapa en ny include-katalog och skapa länkar till andra include-kataloger:

   ```
   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++
   ```

Länkarna kan behöva en del modifikationer, beroende på vilken distribution du använder. Åtminstone Slackware stoppar g++-header-filer i `/usr/local/g++-include`, medan Debian stoppar header-filerna i `/usr/include/g++` och länkar `/usr/lib/g++-include` till `/usr/include/g++`. I det senare fallet gör du nog bäst i att flytta tillbaks den ursprungliga g++-include-katalogen till `/usr/include`.

4. Återställ alla extra header-filer och länkar. Vissa tilläggsbibliotek, som ncurses, stoppar filer i `/usr/include` eller stoppar en länk till sina include-kataloger i `/usr/include`. Dessa filer och länkar måste återställas för att du ska kunna använda dessa bibliotek.

5. Lägg till din nya bibliotekskatalog (som `/usr/i486-linuxlibc5/lib`) *längst upp* i din `/etc/ld.so.conf`-fil. Du bör ha ld.so 1.8.8 eller senare installerad för att inte få konstiga meddelanden när glibc är installerat.

6. Flytta/kopiera all de gamla C-biblioteken till den nya katalogen.

```
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` och `libc.so.5` bör kopieras snarare än flyttas om `/usr` finns på en annan partition än `/`, eftersom de behövs av programmen som används för att starta Linux och måste finnas på root-partitionen.

7. Flytta `/usr/lib/*.o`-filerna till den nya katalogen.

```
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. Uppdatera din biblioteks-cache efter att biblioteken flyttats.

```
ldconfig -v
```

5.3 Installering från binärpaket.

Om du installerar glibc från förkompilerade binärer måste du:

```
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
```

Om du har en annan arkitektur eller version så ska du naturligtvis byta ut filnamnen.

5.4 Installera från källkod.

För att installera biblioteket från källkod, skriv:

```
make install
ldconfig -v
```

5.5 Uppdatera gcc-specs.

Det sista steget i installeringen (för båda binär- och källkods-installering) är att uppdatera gccs `specs`-fil så att du kan länka dina program. För att avgöra vilken specs-fil som gcc använder så kör du:

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

I det här fallet är systemet i486-unknown-linux och 2.7.2.2 är versionen. Du ska kopiera `/usr/lib/gcc-lib/<system>` till den gamla system-katalogen:

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

Gå till den ursprungliga katalogen katalogens versionskatalog:

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

och editera filen `specs` som finns där. I denna fil ska du byta ut `/lib/ld-linux.so.1` till `/lib/ld-linux.so.2`. Du måste också ta bort alla `%{...:-lgmon}`-uttryck i filen, eftersom glibc inte använder gmon-biblioteket för "profiling" (svensk översättning? övers.anm.). Ett exempel på en specs-fil hittar du i avsnittet 9 (Exempel på specs-fil).

5.6 Testa din installering.

För att testa installeringen kan du skapa följande program i filen glibc.c:

```
#include <stdio.h>

main()
{
    printf("hej världen!\n");
}
```

och kompilera programmet.

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

Använd ldd för att se efter så att programmet länkades med glibc2 och inte ditt gamla libc:

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

Om kompileringen går bra och visar "hej världen!" när det körs så var installeringen framgångsrik.

6 Kompilera med det icke-primära libc.

Ibland kommer du vilja använda ett alternativt bibliotek att kompilera dina program med. Detta avsnitt förklarar hur du kan göra detta. Kataloger och installeringsnamn är desamma som i exemplen i de två föregående avsnitten. Kom ihåg att ändra namnen för att passa dina inställningar.

6.1 En varning angående användandet av icke-primära libc.

Innan du kompilerar några program som används i systemets bootprocess måste du tänka på att programmet länkas dynamiskt och används innan icke-rotpartitionerna monteras, så alla länkade bibliotek måste finnas på rotpartitionen. Om du följer installeringsprocessen i föregående avsnitt för att installera glibc som ditt primära C-bibliotek så lämnas det gamla libc i `/lib`, alltså på din rotpartition. Detta innebär att alla dina program fortfarande kommer att fungera då du bootar. Om dock `/usr` finns på en annan partition och du installerar glibc som testbibliotek i `/usr/i486-linuxglibc2`, så kommer inga nya program du kompilerar med glibc fungera förrän din `/usr`-partition monterats.

6.2 Kompilera program med ett test-glibc.

För att kompilera ett program med ett test-installerat glibc måste du definiera om include-sökvägarna, så att de pekar på include-filerna för glibc. Genom att ange "-nostdinc" får du kompilatorn att ignorera den vanliga sökvägen. "-I/usr/i486-linuxglibc2/include" gör att den hittar include-filerna för glibc. Du måste även ange include-filerna för gcc, vilka finns i `/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include` (förutsatt att du installerat test-biblioteket i i486-linuxglibc med gcc 2.7.2.2).

För att länka ett program med ett test-installerat glibc måste du ange korrekt gcc-setup. Detta gör du genom att ange parametern "-b i486-linuxglibc2".

För de flesta program kan de ange dessa nya parametrar genom att lägga till dem till `$CFLAGS` och `$LDFLAGS` i Makefilen:

```
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
```

Om du använder ett configure-script så kan du ange `$CFLAGS` och `$LDFLAGS` som skal-variabler (genom att använda env/setenv för csh/tsch eller set/export för sh/bash osv.) innan du kör configure. Makefilerna som skapas på detta sätt skall innehålla korrekta `$CFLAGS` och `$LDFLAGS`. Inte alla configure-scripts kommer ta hänsyn till skal-variablerna, så du bör kolla och editera Makefilen för hand efter att du kört configure.

Om programmen du kompilerar endast anropar gcc (och inte cpp eller binutils direkt), så kan du använda följande skal-program för att slippa ange parametrarna varje gång:

```
#!/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 "$@"
```

Sedan kan du använda detta skal-program istället för "gcc" när du kompilerar.

6.3 Kompilera program med libc5 när glibc är primärt bibliotek.

För att kompilera ett program med dina gamla bibliotek, när du har installerat glibc som ditt huvudbibliotek, så måste du ställa om include-sökvägen till de gamla include-filerna. Att ange "-nostdinc" gör att de normala sökvägarna negeras, och "-I/usr/i486-linuxlibc5/include" pekar på include-filerna för libc5. Du måste också ange "-I/usr/lib/gcc-lib/i486-linuxlibc5/2.7.2.2/include" för att få med de gcc-specifika include-filerna. Kom ihåg att justera dessa sökvägar i överensstämmelse med namnen på dina nya kataloger och din gcc-version.

För att länka ett program med ditt gamla libc måste du ange gcc-setup. Detta gör du med parametern "-b i486-linuxlibc5".

För de flesta program kan du ange dessa nya alternativ genom att lägga till dem till `$CFLAGS` och `$LDFLAGS` i Makefilen:

```
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
```

Om du använder ett configure-script så kan du ange `$CFLAGS` och `$LDFLAGS` som skal-variabler (genom att använda env/setenv för csh/tsch eller set/export för sh/bash osv.) innan du kör configure. Makefilerna som skapas på detta sätt skall innehålla korrekta `$CFLAGS` och `$LDFLAGS`. Inte alla configure-scripts kommer ta hänsyn till skal-variablerna, så du bör kolla och editera Makefilen för hand efter att du kört configure.

Om programmen du kompilerar endast anropar gcc (och inte cpp eller binutils direkt), så kan du använda följande skal-program för att slippa ange parametrarna varje gång:

```
#!/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 "$@"
```

Sedan kan du använda detta skalprogram istället för "gcc" när du kompilerar.

7 Kompilera C++-program.

Libg++ använder delar av math-biblioteket, så det länkas till libm. Eftersom ditt existerande libg++ kommer kompileras med ditt gamla bibliotek, så måste du kompilera om libg++ med glibc eller skaffa en binärpaket. Den senaste källkoden för libg++, tillsammans med en binär länkad med glibc (för x86) hittar du på `<ftp://ftp.yggdrasil.com/private/hjl/>`.

7.1 Installera libg++ för ett test-installerat glibc.

Om du har installerat glibc som ett test-bibliotek så måste du installera filerna till katalogen du installerade glibc i (som t.ex. `/usr/i486-linuxglibc2` i exemplen i föregående avsnitt).

If du installera från binärpaketet (vilket jag skulle rekommendera, eftersom jag aldrig har lyckats kompilera libg++ på detta sätt), så måste du packa upp filerna till en temporär katalog och flytta alla `usr/lib/`-filer till `<installerings-katalog>/lib/`-katalogen, `usr/include/`-filerna till `<installerings-katalog>/include/`-katalogen (kom ihåg att radera din `include/g++`-länk först!), och sedan `usr/bin/`-filerna till `<installerings-katalog>/bin/`-katalogen.

7.2 Installera libg++ för glibc som primärt bibliotek.

Om du har installerat glibc som ditt primära bibliotek så måste du först flytta dina gamla libg++-filer till din gamla libc-katalog, om du fortfarande vill kunna kompilera g++-program med ditt gamla libc. Det enklaste sättet att göra detta är att installera en ny kopia av libg++, kompilerad med libc5 enligt föregående avsnitt, och sedan installera glibc-versionen på det vanliga sättet.

7.3 Kompilera C++-program med det icke-primära libc.

Om du försöker kompilera C++-program med ett icke-primärt libc så måste du inkludera include-katalogen för g++, vilken i exemplen ovan är `/usr/i486-linuxglibc2/include/g++` (för en test-installering av glibc2) eller `/usr/i486-linuxlibc5/include/g++` (för en installering av glibc2 som primärt bibliotek). Detta kan vanligtvis göras genom att lägga till `$CXXFLAGS`-variabeln:

```
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 Rapportera buggar.

Om du tror att biblioteket är buggigt, läs först FAQen. Det kan vara så att andra har haft samma problem och att det finns någon enkel lösning. Du bör också läsa avsnittet ”Recommended Tools to Install the GNU C Library” i `INSTALL`-filen, eftersom vissa buggar kan finnas i verktygen och inte i glibc.

Om du hittat en bugg, så se till att det verkligen är en bugg. Ett bra sätt att göra detta är att kolla om GNU C-biblioteket uppträder på samma sätt som andra C-bibliotek. Om det gör det så har du antagligen fel och biblioteken har antagligen rätt (men inte med nödvändighet). Om inte så har säkert biblioteken fel.

Gå sedan till `<http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl>` och ta en titt på bugg-databasen. Kolla så att problemet inte redan har rapporterats. Du bör också titta på filen `BUGS` (vilken kommer med libc) för att se efter om det är en känd bugg.

Om du är säker på att du hittat en ny bugg, försök avgränsa den till det minsta möjliga testfallet som producerar detta problem. I C-bibliotekets fall behöver du bara avgränsa det till ett biblioteksfunktionsanrop, om det är möjligt. Detta bör inte vara så svårt.

Det sista steget, när du har ett enkelt testfall är att rapportera buggen. När du rapporterar en bugg, skicka in testfallet, resultaten du fått, resultaten du väntade dig, vad du tror felet kan vara (om du har kommit på något), vilken sorts system du har, versionerna av GNU C-biblioteket, GNU CC-kompilatorn och GNU Binutils som du använder. Inkludera också filerna `config.status` och `config.make`, vilka skapas då du kör `configure`; de kommer finnas i den katalog du stod i då du körde `configure`.

Alla buggrapporter för GNU C-biblioteket ska skickas med hjälp av skalprogrammet `glibcbug`, vilket kommer med GNU libc, till `<bugs@gnu.org>` (den äldre adressen `<bugs@gnu.ai.mit.edu>` fungerar fortfarande), eller skickad genom webb-gränssnittet GNATS till `<http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl>`.

Förslag och frågor ska skickas till mailinglistan på `<bugs-glibc@prep.ai.mit.edu>`. If du inte läser nyhetsgruppen gnu.bug.glibc så kan du prenumerera på listan genom att fråga `<bug-glibc-request@prep.ai.mit.edu>`.

Var vänlig skicka INTE buggrapporter för GNU C-biblioteket till <bug-gcc@prep.ai.mit.edu>. Den listan är till för buggrapporter för GNU CC. GNU CC och GNU C-biblioteket är separata entiteter vilka utvecklas av olika människor.

9 Exempel på specs-fil.

Nedan följer ett exempel på en `specs`-fil för glibc2, vilken används av gcc vid kompilering och länkning. Den bör finnas i katalogen `/usr/lib/gcc-lib/<ny system-katalog>/<gcc-version>`. Om du kör ett x86-system så kan du antagligen kopiera det här avsnittet direkt till filen.

```
*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 Diverse.

10.1 Vidare information.

10.1.1 Webb-sidor.

- *FSF's GNU C Library Home Page* `<http://www.gnu.org/software/libc/libc.html>`
- *Using GNU Libc 2 with Linux* `<http://www.imaxx.net/~thrytis/glibc/>`
- *Installing glibc-2 on Linux* `<http://huizen.dds.nl/~frodol/glibc/>`.
- *Debian libc5 to libc6 Mini-HOWTO* `<http://www.gate.net/~storm/FAQ/libc5-libc6-Mini-HOWTO.html>`.

10.1.2 Nyhetsgrupper.

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

10.1.3 Mailinglistor.

Glibc 2 Linux discussion list.

Denna lista är avsett för diskussion mellan Linux-användare som installerat glibc2, det nya GNU C-biblioteket. Ämnena kan t.ex. vara kompatibilitesproblem och frågor om kompilering av kod i en Linux/glibc-miljö. För att prenumerera, skicka ett ebrev till *Majordomo@ricardo.ecn.wfu.edu* `<mailto:Majordomo@ricardo.ecn.wfu.edu>` med "subscribe glibc-linux <din epostadress> i själva meddelandet.

10.2 Tillkännagivanden.

Det mesta av denna information är stulen från *GNU Libc web page* `<http://www.gnu.org/software/libc/libc.html>` och från Ulrich Dreppers <drepper@gnu.ai.mit.edu> glibc2-kungörelse och hans kommentarer. Andreas Jaeger <aj@arthur.rhein-neckar.de> bidrog med delar av avsnittet om att rapportera buggar.

Följande människor har bidragit med information och respons på detta dokument:

- 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>

Översättningar av detta dokument utförs av:

- Kinesiska: Allex <allex@ms2.accmail.com.tw>
- Franska: Olivier Tharan <tharan@int-evry.fr>
- Japanska: Kazuyuki Okamoto <ikko-@pacific.rim.or.jp>

10.3 Respons.

Förutom att jag skrivit denna HOWTO, håller i webb-sidan *glibc 2 for Linux* `<http://www.imaxx.net/~thrytis/glibc>` och använder glibc på min maskin, så har jag inget att göra med glibc-projektet. Jag är långt ifrån kunnig på området, men jag brukar försöka hjälpa till då folk skicka mig ebrev med frågor. Jag välkomnar alla former av respons, rättelser eller förslag som du har att erbjuda. Skicka dem till *ejg3@cornell.edu* `<mailto:ejg3@cornell.edu>`.

10.4 Upphovsrätt.