Glibc 2 HOWTO

Table of Contents

Table of Contents

Glibc 2 HOWTO

Eric Green, `ejg3@cornell.edu`

v1.6, 22 June 1998

The glibc 2 HOWTO covers installing and using the GNU C Library version 2 (libc 6) on Linux systems.

6.Compiling with the non−primary libc.

7.Compiling C++ programs.

8.Problems.

9.Reporting bugs.

10.Sample specs file.

11.Miscellanea.

1.Introduction.

1.1 About glibc 2.

Glibc 2 is the latest version of the GNU C Library. It currently runs unmodified on GNU Hurd systems and Linux i386, m68k, and alpha systems. Support for Linux PowerPC, MIPS, Sparc, Sparc 64, and Arm will be in version 2.1. In the future support for other architectures and operating systems will be added.

On Linux, glibc 2 is used as the libc with major version 6, the successor of the Linux libc 5. It is intended by the Linux libc developers to eventually replace libc 5. As of 2.0.6, glibc is considered production quality. Version 2.1 (due out in the near future) will be ready for main stream use along with adding more ports and features.

There are three optional add−ons available for glibc 2:

Crypt

The UFC−crypt package. It is seperate because of export restrictions.

LinuxThreads

An implementation of the Posix 1003.1c "pthread" interface.

Locale data

Contains the data needed to build the locale data files to use the internationalization features of the glibc.

The crypt and LinuxThreads add−ons are strongly recommended... not using them risks to be incompatible with the libraries of other systems. (If you do not wish to use them, you must add the option −−disable−sanity−checks when you run configure.)

1.2 About this document.

This HOWTO covers installing the glibc 2 library on an existing Linux system. It is tailored for users of Intel based systems currently using libc 5, but users of other systems and alternate libraries (such as glibc 1) should be able to use this information by substituting the proper filenames and architecture names in the appropriate places.

The latest copy of this HOWTO can be found as part of the Linux Documentation Project or from http://www.imaxx.net/~thrytis/glibc/Glibc2−HOWTO.html.

1.3 Recent changes in this document.

Differences between version 1.6 and 1.5:

- Fixed the install instructions of the binary glibc package.

Differences between version 1.5 and 1.4:

- Indexing added Ed Bailey.

- Changed my email address.

2.Choosing your installation method.

There are a few ways to install glibc. You can install the libraries as a test, using the existing libraries as the default but letting you try the new libraries by using different options when compiling your program. Installing in this way also makes it easy to remove glibc in the future (though any program linked with glibc will no longer work after the libraries are removed). Using glibc as a test library requires you to compile the libraries from source. There is no binary distribution for installing libraries this way. This installation is described in Installing as a test library.

The other way described in this document to install is using glibc as your primary library. All new programs that you compile on your system will use glibc, though you can link programs with your old libraries using different options while compiling. You can either install the libraries from binaries, or compile the library yourself. If you want to change optimization or configuration options, or use an add−on which is not distributed as a binary package, you must get the source distribution and compile. This installation procedure is described in Installing as the primary C library.

Frodo Looijaard describes yet another way of installing glibc. His method involves installing glibc as a secondary library and setting up a cross compiler to compile using glibc. The installation procedure for this method is more complicated then the test library install described in this document, but allows for easier compiling when linking to glibc. This method is described in his Installing glibc−2 on Linux document.

If you are currently running Debian 1.3 but do not want to upgrade to the unstable version of Debian to use glibc, the Debian libc5 to libc6 Mini−HOWTO describes how to use Debian packages to upgrade your system.

If you are installing glibc 2 on an important system, you might want to use the test install. Even if there are no bugs, some programs will need to be modified before they will compile due to changes in function prototypes and types.

3.Getting the library.

The glibc 2 consists of the glibc package and three optional add−on packages, LinuxThreads, Locale, and Crypt. The source can be found at

- 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

It will take about 150 MB of disk space for the full compile and install. The basic binary install of just the core library package is about 50 MB.

Binary packages for 2.0.6 are not available. Version 2.0.4 binary packages are available for i386 and m68k, and version 2.0.1 for the alpha can be found at

- 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

There are export restrictions on the crypt add−on. Non−US users should get it from ftp://ftp.ifi.uio.no/pub/gnu.

If you are running a Red Hat distribution, you can get rpms for glibc 2 from ftp://ftp.redhat.com/pub/redhat/. Glibc 2 is the primary C library for the new Red Hat distribution 5.0.

If you are running a Debian distribution, you can get the packages for glibc 2 from ftp://ftp.debian.org/debian/dists/unstable/main/. The files are named libc6. Glibc 2 is now part of the base package of the hamm version of Debian, and will be the primary libc when Debian 2.0 is released.

4.Installing as a test library.

This section covers installing glibc 2 as a test library. Anything you compile will be linked to your existing libraries unless you give some extra parameters to link to the new libraries. It appears that the paths are compiled into quite a few files, so you probably have to install the library from source.

4.1 Compiling and installing.

Prerequisites.

- About 150 MB free disk space
- GNU make 3.75
- gcc >= 2.7.2 (better 2.7.2.1)
- binutils 2.8.1 (for alpha you need a snapshot)
- bash 2.0
- autoconf 2.12 (if you change configure.in)
- texinfo 3.11

On an i586@133 with 64 MB of RAM, it takes about 3 hours to compile with full libraries with add−ons. On a loaded i686@200, it takes about half an hour.

Extracting the source.

You need to extract the source from the archives so you can compile it. The best way to do this is:

```
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
```

This will put linuxthreads, crypt, and localedata directories in the glibc−2.0.6 directory where configure can find these add−ons.

Configuring.

In the glibc−2.0.6 directory, create a directory named compile, and cd into it. All work will be done in this directory, which will simplify cleaning up. (The developers have not been very concerned with getting 'make clean' perfect yet.)

```
mkdir compile
cd compile
```

Run `../configure`. To use the add−on packages, you need to specify them with −−enable−add−ons, such as −−enable−add−ons=linuxthreads,crypt,localedata. You also need to choose a destination directory to install to. /usr/i486−linuxglibc2 is a good choice. The configure line for this would be:

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

Compiling and installing.

To compile and verify, run:

```
make
make check
```

If the 'make check' succeeds, install the library as root (while still in the compile/ directory):

```
make install
```

4.2 Updating the dynamic loader.

1. Create a link from the new ld.so to /lib/ld-linux.so.2:

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

   This is the only library where the location is fixed once a program is linked, and using a link in /lib will ease upgrading to glibc as your primary C library when the stable version is released.
2. Edit /etc/ld.so.conf. You need to add path to the lib directory the new libraries reside in at the end of the file, which will be <prefix>/lib, such as /usr/i486-linuxglibc2/lib for the choice above. After you have modified /etc/ld.so.conf, run

   ```
   ldconfig -v
   ```

4.3 Configuring for gcc.

The last step of installation is updating /usr/lib/gcc-lib so gcc knows how to use the new libraries. First you need to duplicate the existing configuration. To find out which configuration is current, use the -v option of gcc:

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

In this case, i486−unknown−linux is the system, and 2.7.2.2 is the version. You need to copy the `/usr/lib/gcc-lib/<system>` to the new test system directory:

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

Change into your new test system directory and version directory

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

and edit the file `specs` found in this directory. In this file, change `/lib/ld-linux.so.1` to `/lib/ld-linux.so.2`. You also need to remove all expressions `%{...:-lgmon}` in the file, since glibc does not use the gmon library for profiling. A sample specs file can be found in the Sample specs file section.

4.4 Updating header file links.

You need create links in your new include directory to other include directories:

```
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
```

You might also have other libraries such as ncurses which need their header files put in this directory. You should copy or link the files from `/usr/include`. (Some libraries may need to be recompiled with glibc2 in order to work with it. In these cases, just compile and install the package to `/usr/i486-linuxglibc2`.)

4.5 Testing your installation.

To test the installation, create the following program in a file glibc.c:

```
#include <stdio.h>

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

and compile with the options of "−b <base install directory> −nostdinc −I<install directory>/include −I/usr/lib/gcc−lib/<new system dir>/<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
```

Use ldd to verify the program was linked with glibc2, and not your old 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)
```

If it compiles, the links check out, and it generates "hello world!" when run, the installation succeeded.

5.Installing as the primary C library.

This section covers installing glibc 2 as your primary C library. Any new programs you compile will be linked with this library, unless you use special compile options to link with another version.

If you are are using Redhat or Debian and have downloaded the appropriate rpm or deb files, see the Redhat or Debian installion instructions. You can then skip this section.

5.1 Building the library from source.

This section explains how to compile glibc 2 and add−ons from the sources. You must compile the library if you want to change optimization or configuration options or use a package you do not have the binaries for.

Prerequisites.

- About 150 MB free disk space
- GNU make 3.75
- gcc >= 2.7.2 (better 2.7.2.1)
- binutils 2.8.1 (for alpha you need a snapshot)
- bash 2.0
- autoconf 2.12 (if you change configure.in)

- texinfo 3.11

On an i586@133 with 64 MB of RAM, it takes about 3 hours to compile with full libraries with add−ons. On a loaded i686@200, it takes about half an hour.

Extracting the source.

You need to extract the source from the archives so you can compile it. The best way to do this is:

```
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
```

This will put linuxthreads, crypt, and localedata directories in the glibc−2.0.6 directory where configure can find these add−ons.

Configuring.

In the `glibc-2.0.6` directory, create a directory named compile, and cd into it. All work will be done in this directory, which will simplify cleaning up. (The developers have not been very concerned with getting 'make clean' perfect yet.)

```
mkdir compile
cd compile
```

Run `../configure`. To use the add−on packages, you need to specify them with −−enable−add−ons, such as −−enable−add−ons=linuxthreads,crypt,localedata. You probably will also want to specify paths where it will be installed. To match the standard linux distributions, specify −−prefix=/usr. (When a prefix of `/usr` is specified on a linux system, configure knows to adjust other paths to place libc.so and other important libraries in `/lib`.) The whole configure line would be:

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

Compiling.

To compile and verify, run:

```
make
```

```
make check
```

5.2 Preparing for installation.

Now you need to move some files around to prepare for the new library, whether you are installing from source or binaries. Any new program compiled will be linked to glibc, but old programs which are not statically linked will still depend on libc 5, so you can not just overwrite the old version.

1. Create a new directory to hold the old files to:

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

2. The old header files must be evacuated from `/usr/include`:

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

3. Create a new include directory and set up the links to other include directories:

   ```
   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++
   ```

   The links may need adjusting according to your distribution. At least Slackware puts g++ headers in `/usr/local/g++-include`, while Debian puts the headers in `/usr/include/g++`, and links `/usr/lib/g++-include` to `/usr/include/g++`. In the later case, you probably will want to move the original g++ include directory back to `/usr/include`.
4. Restore any extra header files and links. Some non−standard libraries such as ncurses put files in `/usr/include` or put a link to their include directories in the `/usr/include`. These files and links need to be restored in order to use the extra libraries properly.
5. Add your new library directory (such as `/usr/i486-linuxlibc5/lib`) *at the top* of your `/etc/ld.so.conf` file. You should have ld.so 1.8.8 or better installed to avoid getting strange messages once glibc is installed.
6. Move/copy all the old C libraries into the new directory.

   ```
   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` and `libc.so.5` should be copied and not moved if `/usr` is a seperate partition from `/`, because they are required by programs used to start linux and must be located on the root drive partition.

7. Move the `/usr/lib/*.o` files into the new directory.

```
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. Update your library cache after your libraries are moved.

```
ldconfig -v
```

5.3 Installing from the binary package.

If you are installing glibc from precompiled binaries, you first want to check what is in the package before you install the binaries:

```
tar -tzvvf glibc-2.0.bin.i386.tar.gz
tar -tzvvf glibc-crypt-2.0.bin.i386.tar.gz
```

If you are happy with that, you can install glibc with:

```
cd /
tar -xzf glibc-2.0.bin.i386.tar.gz
tar -xzf glibc-crypt-2.0.bin.i386.tar.gz
ldconfig -v
```

If you have a different architecture or version, substitute the proper file names.

The most recent glibc version is generally not available as a binary package, and it is strongly recommended that you run the most recent version to avoid bugs. If you can not build the library yourself, grab a binary package of glibc from one of the distributions that is based on glibc (e.g. RedHat) and install this.

5.4 Installing from the source.

To install the library from source, run as root from the `compile/` directory:

```
make install
ldconfig -v
```

5.5 Updating the gcc specs.

The final step of the installation (for both binary and source installs) is to update the gcc `specs` file so you can link your programs properly. To determine which specs file is the one used by gcc, use:

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

In this case, i486−unknown−linux is the system, and 2.7.2.2 is the version. You need to copy the `/usr/lib/gcc-lib/<system>` to the old system directory:

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

Change into the original directory and version directory

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

and edit the file `specs` found in this directory. In this file, change `/lib/ld-linux.so.1` to `/lib/ld-linux.so.2`. You also need to remove all expressions `%{...:-lgmon}` in the file, since glibc does not use the gmon library for profiling. A sample specs file can be found in the Sample specs file section.

5.6 Testing your installation.

To test the installation, create the following program in a file glibc.c:

```
#include <stdio.h>

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

and compile the program.

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

Use ldd to verify the program was linked with glibc2, and not your old libc:

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

If this compiles and generates "hello world!" when run, the installation was successful.

6.Compiling with the non−primary libc.

There are times you will want to use an alternate library to compile your programs with. This section explains how to accomplish this, using the directories and installation names used in the examples in the previous two sections. Remember to change the names to fit your setup.

6.1 A warning when using non−primary libcs.

Before compiling any programs which is used in the system boot process, remember that if the program is dynamically linked and is used before the non−root partitions are mounted, all linked libraries must be on the root partition. Following the installation process in the previous section for installing glibc as your primary C library, the old libc is left in `/lib`, which will be on your root partition. This means all of your programs will still work during booting. However, if `/usr` is on a different partition and you install glibc as a test library in `/usr/i486−linuxglibc2`, any new programs you compile with glibc will not work until your `/usr` partition is mounted.

6.2 Compiling programs with a test glibc.

To compile a program with a test−install glibc, you need to reset the include paths to point to the glibc includes. Specifying "−nostdinc" will negate the normal paths, and "−I/usr/i486−linuxglibc2/include" will point to the glibc includes. You will also need to specify the gcc includes, which are found in `/usr/lib/gcc-lib/i486-linuxglibc2/2.7.2.2/include` (assuming you installed the test lib in i486−linuxglibc2 with gcc version 2.7.2.2).

To link a program with a test−install glibc, you need to specify the gcc setup. This is done by using the option "−b i486−linuxglibc2".

For most programs, you can specify these new options by adding them to the `$CFLAGS` and `$LDFLAGS` makefile options:

```
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
```

If you are using a configure script, define the `$CFLAGS` and `$LDFLAGS` shell variables (by using env/setenv for csh/tcsh, or set/export for sh/bash/etc) before running configure. The makefiles generated by this should contain the proper `$CFLAGS` and `$LDFLAGS`. Not all configure scripts will pick up the variables, so you should check after running configure and edit the makefiles by hand if necessary.

If the programs you are compiling only call gcc (and not cpp or binutils directly), you can use the following script to save having to specify all of the options each time:

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

You can then use this script instead of "gcc" when compiling.

6.3 Compiling programs with libc 5 when glibc is primary library.

To compile a program with your old libraries when you have installed glibc as your main library, you need to reset the include paths to the old includes. Specifying "−nostdinc" will negate the normal paths, and "−I/usr/i486−linuxlibc5/include" will point to the glibc includes. You must also specify "−I/usr/lib/gcc−lib/i486−linuxlibc5/2.7.2.2/include" to include the gcc specific includes. Remember to adjust these paths based on the what you named the new directories and your gcc version.

To link a program with your old libc, you need to specify the gcc setup. This is done by using the option "−b i486−linuxlibc5".

For most programs, you can specify these new options by appending them to the `$CFLAGS` and

`$LDFLAGS` makefile options:

```
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
```

If you are using a configure script, define the `$CFLAGS` and `$LDFLAGS` shell variables (by using env/setenv for csh/tcsh, or set/export for sh/bash/etc) before running configure. The makefiles generated by this should contain the proper `$CFLAGS` and `$LDFLAGS`. Not all configure scripts will pick up the variables, so you should check after running configure and edit the makefiles by hand if necessary.

If the programs you are compiling only call gcc (and not cpp or binutils directly), you can use the following script to save having to specify all of the options each time:

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

You can then use this script instead of "gcc" when compiling.

7.Compiling C++ programs.

Libg++ uses parts of the math library, so is link to libm. Since your existing libg++ will be compiled with your old library, you will have to recompile libg++ with glibc or get a binary copy. The latest source for libg++ along with a binary linked with glibc (for x86) can be found at ftp://ftp.yggdrasil.com/private/hjl/.

7.1 Installing libg++ for a test glibc install.

If you have installed glibc as a test library, you need to install the files into the directory you installed glibc into (such as `/usr/i486-linuxglibc2` for the example in the previous sections). If you are installing from the binary package (which i would recommend, since i never had any luck compiling libg++ this way), you need to extract the files into a temporary directory and move all the `usr/lib/` files into the `<install directory>/lib/` directory, the `usr/include/` files into the `<install directory>/include/` directory (remember to delete your `include/g++` link first!), and the `usr/bin/` files into the `<install directory>/bin/` directory.

7.2 Installing libg++ for a primary glibc install.

If you have installed glibc as the primary library, you first need to move your old libg++ files into your old libc directory if you still want to be able to compile g++ programs with your old libc. Probably the easiest way to do this is by installing a new copy of the libg++ compiled with libc 5 as in the previous section, and then installing the glibc version normally.

7.3 Compiling C++ programs with the non−primary libc.

If you are trying to compile a C++ program with a non−primary libc, you will need to include the g++ include dir, which in the examples above would be `/usr/i486−linuxglibc2/include/g++` for a test glibc install or `/usr/i486−linuxlibc5/include/g++` for a primary glibc install. This can usually be done by appending the `$CXXFLAGS` variable:

```
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.Problems.

The glibc package contains a FAQ with additional information that you should check if you are having problems. An online version is also available at http://www.imaxx.net/~thrytis/glibc/glibc−FAQ.html. Below are some tips for solving problems which are not covered in the FAQ or are covered here in more detail.

8.1 Host names do not resolve.

Glibc 2 uses a different method than libc 5 in looking up host names. The glibc name server switch (NSS) code looks for a file `/etc/nsswitch.conf`. If host names are not resolving for you when using a glibc 2 application and your `/etc/resolv.conf` is configured correctly, check if you have the `/etc/nsswitch.conf` file. If you do not have this file, you can create one containing the line:

```
hosts:        files dns
```

It will now look for `/etc/resolv.conf` to find the nameservers.

You should look at the section of the libc info pages describing the nsswitch.conf file for more details.

9.Reporting bugs.

If you think the lib is buggy, please read first the FAQ. It might be that others had the same problem and there's an easy solution. You should also check the section "Recommended Tools to Install the GNU C Library" in the `INSTALL` file since some bugs are bugs of the tools and not of glibc.

Once you've found a bug, make sure it's really a bug. A good way to do this is to see if the GNU C library behaves the same way some other C library does. If so, probably you are wrong and the libraries are right (but not necessarily). If not, one of the libraries is probably wrong.

Next, go to http://www−gnats.gnu.org:8080/cgi−bin/wwwgnats.pl, and look through the bug database. Check here to verify the problem has not already be reported. You should also look at the file `BUGS` (distributed with libc) to check for known bugs.

Once you're sure you've found a new bug, try to narrow it down to the smallest test case that reproduces the problem. In the case of a C library, you really only need to narrow it down to one library function call, if possible. This should not be too difficult.

The final step when you have a simple test case is to report the bug. When reporting a bug, send your test case, the results you got, the results you expected, what you think the problem might be (if you've thought of anything), your system type, the versions of the GNU C library, the GNU CC compiler, and the GNU Binutils which you are using. Also include the files `config.status` and `config.make` which are created by running `configure`; they will be in whatever directory was current when you ran `configure`.

All bug reports for the GNU C library should be sent using the `glibcbug` shell script which comes with the GNU libc to bugs@gnu.org (the older address bugs@gnu.ai.mit.edu is still working), or submitted through the GNATS web interface at http://www−gnats.gnu.org:8080/cgi−bin/wwwgnats.pl.

Suggestions and questions should be sent to the mailing list at bugs−glibc@prep.ai.mit.edu. If you don't read the gnewsgroup gnu.bug.glibc, you can subscribe to the list by asking bug−glibc−request@prep.ai.mit.edu.

Please DO NOT send bug report for the GNU C library to <bug−gcc@prep.ai.mit.edu>. That list is for bug reports for GNU CC. GNU CC and the GNU C library are separate entities maintained by separate people.

10.Sample specs file.

Included here is a sample `specs` file for glibc 2 which is used by gcc for compiling and linking. It should be found in the directory `/usr/lib/gcc-lib/<new system dir>/<gcc version>`. If you are running an x86 system, you probably can copy this section to the file exactly.

```
*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:
. ;
```

11.Miscellanea.

11.1 Further information.

Web pages.

- FSF's GNU C Library Home Page
- Using GNU Libc 2 with Linux
- Installing glibc−2 on Linux.
- Debian libc5 to libc6 Mini−HOWTO.

Newgroups.

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

Mailing lists.

Glibc 2 Linux discussion list.

> This list is intended for discussion among Linux users who have installed glibc2, the new GNU C libraries. Topics might include compatibility issues and questions about the compilation of code in a Linux/glibc setting. To subscribe, send mail to Majordomo@ricardo.ecn.wfu.edu with a body of "subscribe glibc−linux <your email address>.
> Archives for this mailing list can be found at http://www.progressive−comp.com/Lists/?l=linux−glibc&r=1&w=2#linux−glibc

11.2 Credits.

Most of this information was stolen from the GNU Libc web page and from Ulrich Drepper's <drepper@gnu.ai.mit.edu> glibc 2 announcement and his comments. Andreas Jaeger <aj@arthur.rhein−neckar.de> provided some of the Reporting bugs section.

The following people have provided information and feedback for this 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>
- Hank Leininger <hlein@progressive−comp.com>
- 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>
- Michael Wolf <wolfm@rpi.edu>

Translations of this document are being done by:

- Chinese: Allex <allex@ms2.accmail.com.tw>
- French: Olivier Tharan <tharan@int−evry.fr>
- Japanese: Kazuyuki Okamoto <ikko−@pacific.rim.or.jp>

11.3 Feedback.

Besides writing this HOWTO, maintaining the glibc 2 for Linux page, and using it on my machine, I have nothing to do with the glibc project. I am far from knowledgeable on this topic, though I try to help with problems mailed to me. I welcome any feedback, corrections, or suggestions you have to offer. Please send them to ejg3@cornell.edu.

11.4 Copyright.