Glibc 2 하우투

Eric Green, thrytis@imaxx.net v1.2, 26 October 1997

번역: 모름

Glibc 2 하우투는 리눅스 시스템에서 GNU C Library version 2(libc 6)를 설치하고 사용하는 방법을 다룬다.

목 차

제 1 절 소개(Introduction)

1.1 glibc 2란?

Glibc 2은 GNU C Library의 최신 버전이며, 현재 GNU Hurd 시스템, 리눅스 i386, m68k, and alpha 시스템에서 실행된다. 그리고 리눅스 PowerPC, MIPS, Sparc로 포팅작업은 현재 개발 중이다. 앞으로 계획은 다른 architectures와 운영체계에 대한 지원여부를 첨가하는 것이다.^^

리눅스에서는 glibc 2가 Linux libc 5의 다음 버전인 libc major version 6 으로 사용되고 있다. 이러한 과정은 Linux libc 개발자들의 의도이기도 했다. 현재는 실험적이지만 glibc 2에 관심있는 사람들이 평가하고 사용하는데에는 충분히 안정성이 있다. 만약 당신이 사용하는 프로그램이 glibc 2를 지원한다면 최신 버전은 상당히 안정하다. 버전 2.1은 main stream use에 대한 버전이 될 것이다.

glibc 2에서 유용하는 3가지 추가 옵션(optional add-ons)이 있다.

Crypt

UFC-crypt 패키지이다. Export restriction때문에 분리시켰다.

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.

1.2 이 문서에 대해서

이 하우투 문서는 존재하는 리눅스 시스템에서 glibc 2 library를 설치하는 것을 다룰려고 한다. 현재 libc 5를 사용하는 Intel 기반의 시스템의 사용자를 위해서 적었다. 그러나 다른 시스템을 사용하는 사람이나 glibc 1과 같은 변경된 라이브러리를 사용하는 사람도 적절한 위치에 있는 적합한 파일이름과 architecture 이름으로 바꾸는 작업만으로 이 문서의 정보를 사용할 수 있다.

이 하우투 문서의 최신 판은 *Linux Documentation Project* <http://sunsite.unc.edu/LDP> 의 일부로 얻을 수 있거나, <http://www.imaxx.net/~thrytis/glibc/Glibc2-HOWTO.html> 위 위치에서 얻을 수 있다.

1.3 이 문서에서 최근 변경된 내용

버전 1.1과 1.2와 다른 점:

- Added some information to the Reporting bugs section, and updated bug reporting email and gnats addresses. (Changes by Andreas Jaeger)
- Updated Credits section.

버전 1.0과 1.1와 다른 점:

- Updated glibc version to 2.0.5c.
- Added a reference to the Debian libc5 to libc6 하우투 and Frodo Looijaard's Installing glibc-2 on Linux document.

제 2 절 설치하는 방법을 선택하자.

glibc를 설치하는 방법은 많지 않다. 당신도 시험삼아 라이브러리를 설치할수 있다. 즉 기본적으로 이미 설치되어 있는 라이브러리를 사용하여서 glibc 2를 설치한 후, 프로그램을 컴파일 할 때, 다른 옵션을 사용하면 새로운 라이브러리(glibc 2)를 이용할 수 있다. 물론 이런 방법으로 설치하는 것은 설치 후 glibc를 제거하는 것은 쉽다.– 주의할 것은 glibc와 링킹되어서 만들어진 프로그램은 더 이상 실행되지 않는다는 것이다.– glibc를 실험삼아 사용할려면 소스(source)를 가지로 라이브러리를 컴파일하면 된다. 이런 설치 방법은 4(시험삼아 설치하기)에서 자세히 설명했다.

또 다른 설치 방법은 주 라이브러리로 glibc를 사용하기 위한 방법이다. 이것은 당신이 프로그램을 컴파일하면, 새로운 프로그램들은 주로 glibc에 링킹이 된다. 그러나 컴파일 시, 다른 옵션을 사용하면 구 라이브러리에 링킹시킬 수 있다. 바이너리 형태로 라이브러리를 설치할 수 있으며, 또는 스스로 라이브러리를 컴파일 할 수 있다. 만약 최적화 옵션이나 구성 옵션을 변경시키고 싶거나 아니면 바이너리 패키지에서 제공되지 않는 add-on을 사용하려고 한다면, 소스 배포판을 구한 후, 컴파일 하야한다. 이런 설치 과정은 5(primary C library로 설치하기)에서 설명했다.

Frodo Looijaard은 glibc를 설치할 수 있는 다른 방법도 설명했다. 즉, glibc를 secondary libary로 설치하고, glibc를 사용하여서 컴파일되게 cross compiler를 설정하는 방법이다. 이 방법에 대한 설치 과정은 다소 복잡하지만, glibc로 링킹할 때는 쉬운 컴파일을 할 수 있다.(??) 이 방법은 그의 site에 있는 *Installing glibc-2 on Linux* <http://huizen.dds.nl/~frodol/glibc/> 문서에 설명되어있다.

만약 현재 Debian 1.3을 사용 중이고, glibc를 사용하는 데비안의 unstable version 으로 업그레이드를 하기 싫다면, *Debian libc5 to libc6 Mini-HOWTO* `<http://www.gate.net/~storm/FAQ/libc5-libc6-Mini-HOWTO.html>` 을 참고하여서 데비안 패키지를 업그레이드를 시킬 수 있다.

glibc 2를 중요한 시스템에 설치하려고 한다면, 주 라이브러리로 설치하지 말라. 시험삼아 설치하거나, 더 좋은 것은 여유있는 시스템에 설치해서 테스트해 보아라. 설령 버그가 없다하더라도, 어떤 프로그램은 컴파일하기 전에 변경될 필요가 있는데, 이것은 함수의 prototypes과 types이 변경되기 때문이다.

제 3 절 library를 얻기

glibc 2 는 glibc 패키지와 3개의 추가된 옵션 패키지(LinuxThreads, Locale, Crypt)로 구성되어 있다. 소스는 다음 위치에서 찾을 수 있다.

- `<ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.5.tar.gz>`
- `<ftp://prep.ai.mit.edu/pub/gnu/glibc-linuxthreads-2.0.5.tar.gz>`
- `<ftp://prep.ai.mit.edu/pub/gnu/glibc-localedata-2.0.5.tar.gz>`
- `<ftp://prep.ai.mit.edu/pub/gnu/glibc-crypt-2.0.5.tar.gz>`

버전 2.0.5c은 추가적으로 패치가 필요하는 데, 이것은 다음 위치에서 찾을 수 있다.

`<ftp://prep.ai.mit.edu/pub/gnu/glibc-2.0.5-2.0.5c.diff.gz>`.

완전한 컴파일과 설치에 필요한 디스크 공간은 약 150 MB가 필요하다. 주요 라이브러리 패키지만의 기본적인 바이너리 설치시에는 약 50 MB가 필요하다.

버전 2.0.5c에 대한 바이너리 패키지는 구할수 없지만, 버전 2.0.5 바이너리 패키지는 i386, alpha, 68k용으로 배포되며, 다음 위치에서 얻을 수 있다.

- 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 add-on에 대해서는 export restrictions이 있어서, non-US 사용자는 다음 위치에서 crypt add-on 패키지를 얻을 수 있다. `<ftp://ftp.ifi.uio.no/pub/gnu>`.

Red Hat 배포판을 사용하는 사람은 glibc 2용 rpm을 다음 위치에서 얻을 수 있다. `<ftp://ftp.redhat.com/pub/redhat/tbird/RedHat/RPMS/>` 신 베파 배포판인 Red Hat 4.8은 glibc 2가 주 C 라이브러리로 사용되었다.

데비안 배포판을 사용하는 사람은 glibc 2의 패키지를 다음 위치에서 얻을 수 있다. `<ftp://ftp.debian.org/debian/unstable/binary-i386/devel/>`, `<ftp://ftp.debian.org/debian/unstable/binary-m68k/devel/>`, `<ftp://ftp.debian.org/debian/unstable/binary-alpha/devel/>`. 파일은 libc6으로 이름이 되었다. Glibc 2는 데비안의 hamm 버전에서는 기본 패키지의 일부이며, 데비안 2.0부터는 주 libc가 될 것이다.

제 4 절 시험삼아 설치하기

Glibc 2를 시험삼아 설치하면, 컴파일 할 때에 특별한 변수를 사용하여 glibc 2에 링킹시키지 않는한 이미 설치되어 있는 기존의 라이브러리에 링킹이 된다. 설치 디렉토리 위치(path)가 몇 파일에 포함되어 컴파일되므로 , 소스로 부터 라이브러리를 설치해야만 한다.

4.1 컴파일하기와 설치하기

4.1.1 전제 조건

- 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)

RAM 64 MB인 i586@133 컴퓨터에서 추가 라이브러리(add-ons)포함한 전체 라이브러리 를 컴파일하는데 소요 시간은 약 3시간이며, i686@2에 컴퓨터의 경우는 약 1시간 30분 정도 걸렸다.

4.1.2 소스를 추출하기

소스 파일을 컴파일하기 위해서는 Archives에서 소스 파일을 추출해야만 한다. 아래와 같이 하면 된다:

```
tar xzf glibc-2.0.5.tar.gz
cd glibc-2.0.5
cat ../glibc-2.0.5-2.0.5c.diff.gz | gzip -d | patch -p0
tar xzf ../glibc-linuxthreads-2.0.5.tar.gz
tar xzf ../glibc-crypt-2.0.5.tar.gz
tar xzf ../glibc-localedata-2.0.5.tar.gz
```

위의 방법처럼 하면, glibc-2.0.5 디렉토리에 linuxthreads, crypt, localeddata 디렉토리가 생성된다. 이렇게 해야지만 configure가 추가된 라이브러리 디렉토리를 찾을 수 있다.

4.1.3 Configure를 실행하기

glibc-2.0.5 디렉토리에서 "compile"이란 디렉토리를 만들고, cd로 'compile' 디렉토리로 온다. 모든 작업은 이 compile 디렉토리에서 이루어진다.

```
mkdir compile
cd compile
```

위의 과정을 마쳤으면, '../configure'을 실행시키면 된다. 추가 패키지를 사용하고 싶으면, 추가 패키지에 대한 옵션(–enable-add-ons <예> –enable-add-ons=linuxthreads,crypt,localedata)을 포함시켜야 한다. 또 설치하고 싶은 디렉토리를 선택해야 한다. 보통 설치 디렉토리로 '/usr/i486-linuxglibc2'을 사용한다. 다음은 configure line의 예이다:

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

4.1.4 컴파일하기와 설치하기

컴파일하고 제대로 되었는지 확인하는 절차는 다음과 같이 한다:

```
make
make check
```

'make check'가 성공적으로 끝났으면, 라이브러리를 설치한다:

```
make install
```

4.2 동적로더(dynamic loader)를 업그레이드하자.

1. 만들어진 ld.so를 /lib/ld-linux.so.2로 링킹시킨다:

   ```
   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. /etc/ld.so.conf 파일을 편집한다. 파일의 마지막 라인에 새롭게 만들어진 라이브러리 디렉토리의 경로명(path)를 적으면 된다. 즉 우리가 설치했던 경우에는 '/usr/i486-linuxglibc2/lib'을 적으면 된다. (주의사항 : '은 적는 것이 아님.^^) /etc/ld.so.conf 파일을 변경했으면, 다음과 같이 실행해 보라.

   ```
   ldconfig -v
   ```

4.3 gcc를 configure하기

설치의 마지막 단계가 /usr/lib/gcc-lib를 업데이트(update)하는 것이다. 이것을 해야지만 gcc가 어떻게 새로운 라이브러리를 사용할지를 알 수 있다. 우선 있는 configuration파일을 복사한다. 현재 configuration의 내용을 알기 위해서는 다음과 같이 하면 된다:

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

이 경우는 시스템이 i486-unknown-linux, 2.7.2.2가 버전이다. 당신이 해야할 일은 /usr/lib/gcc-lib/<system>을 새로운 테스트 시스템 디렉토리로 복사하는 것이다: 복사하는 것이다.:

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

그 다음 다음과 같이 하라.

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

이 디렉토리에서 'specs'파일을 편집한다. 즉 specs 파일내에 /lib/ld-linux.so.1을 /lib/ld-linux.so.2로 바꾼다. 또 `%{...:-lgmon}` 가 들어간 모든 문장을 지워라. 왜냐면 glibc는 profile하기 위해서 gmon 라이브러리를 사용하지 않기 때문이다. 샘플 specs 파일은 9(샘플 specs 파일)단원 에서 찾을 수 있다.

4.4 헤더 파일 링크를 업데이트하기.

헤더 파일 링크는 다음과 같이 한다:

```
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 설치된것을 테스트하기

우선 다음과 같은 파일(glibc.c)를 만든다:

```
#include <stdio.h>

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

그리고 옵션"-b <base install directory> -nostdinc -I<install directory>/include -I/usr/lib/gcc-lib/<new system dir>/<gcc version>/include" 을 가지고 컴파일 한다.

```
% 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
```

ldd를 이용하여서 프로그램이 glibc2에 링크되었는지 확인해본다:

```
% ldd glibc
libc.so.6 => /usr/i486-linuxglibc2/lib/libc-2.0.5.so (0x4000d000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

컴파일이 되고, 링크 과정도 정상이며, 실행시 "hello world!" 가 출력이 된다면, 설치는 성공적으로 끝난 것이다. (축하....^^)

제 5 절 Primary C library로 설치하기

Glibc 2를 주 C 라이브로리로 설치하는 방법을 설명하겠다. 주 C 라이브러리로 설치 한다는 것은 당신이 컴파일하는 프로그램을 glibc로 링크되게 하는 것이다. 물론, 다른 버전에 링크시킬려면 컴파일 옵션을 추가시켜서 컴파일하면 된다.

만약 당신이 레드헷이나 데비안을 사용하고 적절한 rpm 파일이나 deb 파일을 다운했었다면, 레드헷이나 데비안의 설치 안내를 읽어 보아라. 그러면, 여기있는 내용을 읽지 않아도 된다.

5.1 소스로부터 라이브러리를 설치하기

여기에서는 소스로부터 glibc 2와 add-ons를 컴파일하는 방법을 설명하려고 한다. 최적화 옵션이나 구성 옵션을 변경하고 싶거나 바이너리에 없는 패키지를 사용하려고 할 때 라이브러리를 컴파일할 필요가 있다.

5.1.1 전제 조건

- 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)

RAM 64 MB인 i586@133 컴퓨터에서 추가 라이브러리(add-ons)포함한 전체 라이브러리 를 컴파일하는데 소요 시간은 약 3시간이며, i686@2에 컴퓨터의 경우는 약 1시간 30분 정도 걸렸다.

5.1.2 소스를 추출하기

소스 파일을 컴파일하기 위해서는 Archives에서 소스 파일을 추출해야만 한다. 아래와 같이 하면 된다:

```
tar xzf glibc-2.0.5.tar.gz
cd glibc-2.0.5
cat ../glibc-2.0.5-2.0.5c.diff.gz | gzip -d | patch -p0
tar xzf ../glibc-linuxthreads-2.0.5.tar.gz
tar xzf ../glibc-crypt-2.0.5.tar.gz
tar xzf ../glibc-localedata-2.0.5.tar.gz
```

위의 방법처럼 하면, glibc-2.0.5 디렉토리에 linuxthreads, crypt, localeddata 디렉토리가 생성된다. 이렇게 해야지만 configure가 추가된 라이브러리 디렉토리를 찾을 수 있다.

5.1.3 Configure를 실행하기

glibc-2.0.5 디렉토리에서 "compile"이란 디렉토리를 만들고, cd로 'compile' 디렉토리로 온다. 모든 작업은 이 compile 디렉토리에서 이루어진다.

```
mkdir compile
cd compile
```

위의 과정을 마쳤으면, '../configure'을 실행시키면 된다. 추가 패키지를 사용하고 싶으면, 추가 패키지에 대한 옵션(–enable-add-ons <예> –enable-add-ons=linuxthreads,crypt,localedata)을 포함시켜야 한다. 또 설치하고 싶은 디렉토리를 선택해야 한다. 보통 설치 디렉토리로 '–prefix=/usr'을 사용한다. 다음은 configure line의 예이다:

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

5.1.4 컴파일하기

컴파일하고 제대로 되었는지 확인하는 절차는 다음과 같이 한다:

```
make
make check
```

5.2 설치하기 위한 준비 작업하기

소스로부터 설치했든, 바이너리 파일로 설치했든 glibc 2라이브러리 설치 준비하기 위해선 어떤 파일을 옮길 필요가 있다. 이유는 앞으로 컴파일 할 새로운 프로그램을 glibc에 링크시키고, 정적으로 링크되지 않는 구 프로그램은 여전히 libc 5에 의존 하게 하기위해서이다. 고로 구 버전에 덮어 쓰지 않아야한다.

1. 구 파일(libc 5)을 저장할 새로운 디렉토리를 만든다:

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

2. 구(libc 5용) 헤더 파일을 /usr/include 에서 다른 디렉토리로 옮긴다:

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

3. 새로운(glibc 2용) include 디렉토리를 만들고, 다른 include에 링크시킨다:

   ```
   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++
   ```

   링크는 배포판에 따라서 약간 다를수 있다. 적어도 슬렉웨어의 경우 /usr/local/g++-include에 g++ 헤더파일이 있고, 반면에 데비안 패키지 경우 /usr/include/g++에 g++헤더파일이 있고, /usr/lib/g++-include는 /usr/include/g++ 에 링크되어 있다. 후자의 경우, 아마 당신은 원래 g++ include 디렉토리를 /usr/include에 옮기길 원할 것이다.

4. 다른 헤더 파일을 복원하고 링크한다. 어떤 비표준 라이브러리경우, 예로 ncureses, /usr/include에 헤더 파일을 놓거나 /usr/include 밑에 있는 그들의 include 디렉토리에 링크를 한다. 이 파일들이나 링크는 다른 라이브러리를 적절히 사용될수 있게 복원을 해 둘 필요가 있다.

5. /etc/ld.so.conf파일 *맨위*에 glibc 2용 라이브러리 디렉토리(예, / usr/i486-linuxlibr/lib)를 추가한다. glibc를 설치할 때 이상한 메세지를 피하기 위해서는 ld.so-1.8.8이상의 버전을 설치되어 있어야 한다.

6. 구 C 라이브러리를 새로운 디렉토리(/usr/i486-linuxlibc5/lib)로 옮기거나 복사한다.

```
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
```

만약 /usr이 /와 다른 파티션에 있다면, libm.so.5와 libc.so.5는 move가 아닌 복사되어야 한다. 이유는 리눅스를 시작하는데 필요한 프로그램이 libm.so.5와 libc.so.5를 사용하기때문이고 이것들은 루트 디렉토리에 있어야만 한다.

7. /usr/lib/*.o 파일을 새로운 디렉토리로 옮긴다.

```
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. 당신의 라이브러리를 옮겼으면 라이브러리 cache를 업데이트한다.

```
ldconfig -v
```

5.3 바이너리 패키지로부터 설치하기

미리 컴파일된 바이너리로부터 glibc를 설치하고자 하면, 다음과 같이 한다:

```
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
```

다른 architecture나 다른 버전을 가지고 있다면, 적절한 파일 이름으로 바꾸면 된다.

5.4 소스로부터 설치하기

소스로부터 라이브러리를 설치하려면, 다음을 실행하라:

```
make install
ldconfig -v
```

5.5 gcc specs 을 업데이트하기

바이너리로부터 설치이든 소스로부터 설치하든 마지막 단계는 gcc specs 파일을 업데이트하는 것이다. 그래야만 당신 프로그램이 적절하게 링크될 수 있다. gcc가 사용하는 specs 파일을 알아내기 위해서는 다음과 같이 한다:

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

이 경우는 i486-unknown-linux 시스템이고 버전이 2.7.2.2이다. 당신이 해야할 일은 /usr/lib/gcc-lib/<system>을 구 시스템 디렉토리로 복사하는 것이다:

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

그 다음 다음과 같이 하라.

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

이 디렉토리에서 'specs'파일을 편집한다. 즉 specs 파일내에 /lib/ld-linux.so.1을 /lib/ld-linux.so.2로 바꾼다. 또 `%{...:-lgmon}` 가 들어간 모든 문장을 지워라. 왜냐면 glibc는 profile하기 위해서 gmon 라이브러리를 사용하지 않기 때문이다. 샘플 specs 파일은 9(샘플 specs 파일)단원에서 찾을 수 있다.

5.6 설치된것을 테스트하기

우선 다음과 같은 파일(glibc.c)를 만든다:

```
#include <stdio.h>

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

그리고 컴파일한다.

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

ldd를 이용하여서 프로그램이 glibc2에 링크되었는지 확인해본다:

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

컴파일이 되고, 링크 과정도 정상이며, 실행시 "hello world!" 가 출력이 된다면, 설치는 성공적으로 끝난 것이다. (축하....^^)

제 6 절 non-primary libc로 컴파일하기

프로그램을 컴파일 하다 보면, 다른 라이브러리를 사용하고 싶은 때가 있을 것이다. 이 단원에서는 어떻게 하면 다른 라이브러리를 사용할 수 있는가를 설명한다. 우선 기억해야 할 것은 앞의 두 단원에서 예로 사용되었던 디렉토리 이름과 설치 이름을 사용한다는 것이다. Remember to change the names to fit your setup.

6.1 non-primary libcs 사용할 때 경고

시스템 부팅 과정에 사용되는 프로그램을 컴파일 할 때 기억할 것은, 그 프로그램을 동적으로 링크하고 non-root 파티션이 마운트되기 이전에 사용한다면, 모든 링크되는 라이브러리는 root 파티션에 있어야만 한다는 것이다. glibc를 주 C 라이브러리로 설치하는 과정(앞 단원)을 했다면, 구 libc는 /lib에 있다. /lib는 root 파티션에 있어야한다. 이것은 아직도 모든 당신 프로그램이 부팅하는 동안에는 작동한다는 것이다. 그러나, 만약 /usr이 다른 파티션에 있고, glibc를 /usr/i486-linuxglibc2에 연습용으로 설치했다면, glibc와 함께 컴파일된 프로그램들은 /usr 파티션이 마운트되기 이전까지는 작동하지 않는다는 것이다.

6.2 연습용 glibc와 함께 컴파일하기.

연습용 glibc와 함께 프로그램을 컴파일하기위해선 glibc include에 include path 를 다시 설정할 필요가 있다.

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.

제 7 절 C++ 프로그램을 컴파일하기

libg++는 수학 라이브러리의 일부를 사용하므로 libm을 링크시켜야한다. 기존의 libg++는 구 라이브러리와 함께 컴파되었기때문에, glibc와 다시 컴파일 하거나 아니면 바이너리를 얻어야한다. glibc에 링크된 바이너리 libg++는 다음 위치에서 찾을 수 있다. <ftp://ftp.yggdrasil.com/private/hjl/>.

7.1 연습용 glibc 설치의 경우에 대한 libg++설치하기

연습용으로 glibc를 설치했다면, libg++파일을 glibc가 설치되었던 디렉토리에 설치해야한다. (앞의 단원대로 했다면, /usr/i486-linuxglibc2에 libg++파일을 설치한다.) 바이너리 패키지로부터 설치하고자 한다면, 임시 디렉토리에서 파일내용을 추출한 다음 usr/lib/파일들을 <install directory>/lib/ 디렉토리로 옮기고, usr/include/파일들을 <install directory>/include/ 디렉토리로 옮긴다. (주의 사항: include/g++ 링크를 먼저 지워라), 그리고 usr/bin/파일을 <install directory>/bin/ 디렉토리로 옮긴다.

7.2 주 glibc 설치의 경우에 대한 libg++ 설치하기

주 라이브러리로 glibc를 설치했다면, 먼저 구 libg++ 파일들을 구 libc 디렉토리로 옮겨라. (g++프로그램을 구 libc 와 컴파일하고 싶다면,...) 이것을 하는데 가장 쉬운 방법은 libc 5로 컴파일된 libg++의 새로운 복사판을 설치하고(앞의 단원에서 설명했다), 정상적으로 glibc 버전을 설치한다.

7.3 non-primary libc로 C++프로그램 컴파일하기

C++ 프로그램을 non-primary libc와 컴파일하고 싶으면, g++ include 디렉토리를 포함시켜야한다. (즉 연습용 설치 경우, /usr/i486-linuxglibc2/include/g++이고, 주 glibc 설치 경우는 /usr/i486-linuxlibc5/include/g++이다) 보통 CXXFLAGS 변수를 첨가하는 방법으로 사용할 수 있다.

```
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
```

제 8 절 버그 보고하기

lib에 버그가 있다고 생각되면, 우선 FAQ를 읽어보라. 다른 사람들이 가진 문제점과 같은 경우라면 해결책이 있다. 또는 INSTALL 파일에 있는 "Recommended Tools to Install the GNU C Library" 단원을 체크해 보아라. 이유는 어떤 버그는 glibc의 버그가 아닌 tools의 버그이기때문이다.

일단 버그를 발견했으면, 그것이 진짜 버그인지 확인하라. GNU C 라이브러리가 다른 C 라이브러리와 같은 방법으로 작동하는지 확인하면 된다. 만약 다른 C라이브러 리와 같은 방법으로 작동하지 않으면 라이브러리에 이상이 있는 것이다.

다음은 <http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl>로 접속한 후, 버그 데이타베이스를 검토해 보라. 이미 보고되었던 문제인지 확인해라. libc와 함께 배포되는 BUGS 파일도 보아라.

새로운 버그라고 생각이 되면, 그 문제가 일어날수 있는 경우를 좁힐 수 있을 만큼 좁혀라.(?) C 라이브러리 경우 한 라이브러리 함수 호출로 줄여서 보고할수 있으면 그렇게 해라. 이것이 너무 어렵지 않아야 한다.

간단한 테스트 판을 가지고 있다면 버그를 보고하라. 다음은 버그 보고할 때 포함해야 할 내용이다. (1) 버그가 생기는 test case와 결과들. (2) 만약 버그 원인에 대한 생각이 있으면, 당신의 생각 (3) 당신이 사용했던 시스템 종류, GNU C 라이브러리 버전,GCC CC 컴파일러 버전, GNU Binutils 버전 (4) 'configure'를 실행시켰을 때 만들어지는 'config.status'와 'config.make'의 파일

GNU C 라이브러리에 대한 버그 레포트는 glibcbug 쉘 스크립트를 사용해서 <bugs@gnu.org> 로 보내라. (다른 주소는 <bugs@gnu.ai.mit.edu>) 또는 <http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl>에 있는 GNATS 웹 인터페이스를 통해서 제출할 수 있다.

제안과 질문은 <bugs-glibc@prep.ai.mit.edu>에 있는 메일링 리스트로 보내라. 만약 gnewsgroup인 gnu.bug.glibc를 읽어보지 않았다면, <bug-glibc-request@prep.ai.mit.edu>에 물어서 리스트에 제출할 수 있다.

<bug-gcc@prep.ai.mit.edu> 로 GNU C 라이브러리에 대한 버그 보고서를 보내지 마라. 그곳은 GNU CC에 대한 버그 보고서를 위한 곳이다. GNU CC와 GNU C 라이브러리는 다른 사람들에 의해서 관리된다.

제 9 절 샘플 specs 파일

여기에 있는 샘플 specs파일은 glibc 2용이며 컴파일할 때, 링크할 때 gcc에 의해서 사용되는 파일이다. 이 파일은 /usr/lib/gcc-lib/<new system dir>/<gcc version>에 있다. 만약 x86 시스템을 사용한다면, 이 샘플 specs 파일을 복사할 수 있다.

```
*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:
. ;
```

제 10 절 Miscellanea

10.1 Further information

10.1.1 Web Pages

- *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 Newgroups

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

10.2 Credits

Most of this information was stolen from the *GNU Libc web page* `<http://www.gnu.org/software/libc/libc.html>` 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:

- 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>
- Stephane <sr@adb.fr>
- Jan Vandenbos <jan@imaxx.net>

Translations of this document are being done by:

- French: Olivier Tharan <tharan@int-evry.fr>
- Japanese: Kazuyuki Okamoto <ikko-@pacific.rim.or.jp>

10.3 Feedback

Besides writing this HOWTO, maintaining the *glibc 2 for Linux* `<http://www.imaxx.net/~thrytis/glibc>` 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 *thrytis@imaxx.net* `<mailto:thrytis@imaxx.net>`.