安装 GCC-8.3.0 及其依赖

主要参考：https://www.cnblogs.com/aquester/p/10799125.html

目录

1. 前言

为体验 C++17 和 C++20 特性，需安装更新版本的 GCC 编译器。GCC 官网为：https://gcc.gnu.org/，从这里可以下载最新版本的 GCC。

C++由 Bjarne Stroustrup（被誉为 C++之父）于 1979 年在新泽西州美利山贝尔实验室开始设计开发的，最初命名为带类的 C，后来在 1983 年更名为 C++。

2. 安装日期

2019/4/27，截至该日期最新版本为 GCC-8.3.0，但在本月末或下月初即将发布 GCC-9.1（已于 2019-05-03 发布）。

3. GCC 国内镜像下载地址

下载速度不一，请选择速度最快的：

1) http://mirrors.nju.edu.cn/gnu/gcc/gcc-8.3.0/

2) http://mirrors.ustc.edu.cn/gnu/gcc/gcc-8.3.0/

3) https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-8.3.0/

4. GCC 的依赖库

编译之前需先安装好 GCC 的依赖库：gmp、mpfr 和 mpc。编译还依赖 m4 等编译工具，如果没有，则在执行 configue 时会报相应的错误，这时需要先安装好这些编译工具。

4.1. gmp 库

GMP 为“GNU MP Bignum Library”的缩写，是一个 GNU 开源数学运算库。本文选择的是最新版本 gmp-6.1.2，国内镜像下载地址：

1) https://mirrors.tuna.tsinghua.edu.cn/gnu/gmp/

2) http://mirrors.nju.edu.cn/gnu/gmp/

3) http://mirrors.ustc.edu.cn/gnu/gmp/

4.2. mpfr 库

mpfr 是一个 GNU 开源大数运算库，它依赖 gmp。本文选择的是最新版本 mpfr-4.0.2，国内镜像下载地址：

1) https://mirrors.tuna.tsinghua.edu.cn/gnu/mpfr/

2) http://mirrors.nju.edu.cn/gnu/mpfr/

3) http://mirrors.ustc.edu.cn/gnu/mpfr/

4.3. mpc 库

mpc 是 GNU 的开源复杂数字算法，它依赖 gmp 和 mpfr。本文选择的是最新版本 mpc-1.1.0，国内镜像下载地址：

1) https://mirrors.tuna.tsinghua.edu.cn/gnu/mpc/

2) http://mirrors.nju.edu.cn/gnu/mpc/

3) http://mirrors.ustc.edu.cn/gnu/mpc/

4.4. m4 编译工具

本文选择的是最新版本 m4-1.4.16，下载地址：

1) https://mirrors.tuna.tsinghua.edu.cn/gnu/m4/

2) http://mirrors.nju.edu.cn/gnu/m4/

3) http://mirrors.ustc.edu.cn/gnu/m4/

如果使用“--prefix”指定了安装目录，则在编译 gmp 等之前还需先设置好环境变量 PATH，以便 configure 时能找到 m4。

4.5. 安装源代码包

涉及到的所有安装源代码包：

```
gcc-8.3.0.tar.gz
mpfr-4.0.2.tar.gz
gmp-6.1.2.tar.xz
mpc-1.0.3.tar.gz
```

GCC 的依赖库间还互有依赖：mpfr 依赖 gmp、mpc 依赖 gmp 和 mpfr，所以 GCC 的编译安装顺序为：

1) m4（如果需要）

2) gmp

3) mpfr

4) mpc

5) GCC

为了不污染已有的编译和运行环境，将 GCC 及依赖库均安装到/usr/local 目录，并且最好以 root 用户完成下述所有操作。

5. 编译安装 gmp

执行 configure 生成 Makefile 时，需要用到 m4，一般路径为/usr/bin/m4，如果没有则需要先安装，否则报错“no usable m4”错误，手工安装 m4 从“https://www.gnu.org/software/m4/”下载。

具体安装步骤如下：

```
xz -d gmp-6.1.2.tar.xz
tar xf gmp-6.1.2.tar
cd gmp-6.1.2
./configure --prefix=/usr/local/gmp-6.1.2
make
make install
ln -s /usr/local/gmp-6.1.2 /usr/local/gmp
```

6. 编译安装 mpfr

详细安装步骤如下：

```
tar xzf  mpfr-4.0.2.tar.gz
cd mpfr-4.0.2
./configure --prefix=/usr/local/mpfr-4.0.2 --with-gmp=/usr/local/gmp
make
make install
ln -s /usr/local/mpfr-4.0.2 /usr/local/mpfr
```

7. 编译安装 mpc

```
tar xzf  mpc-1.1.0.tar.gz

cd mpc-1.1.0

./configure --prefix=/usr/local/mpc-1.1.0 --with-gmp=/usr/local/gmp --with-mpfr=/usr/local/mpfr

make

make install

ln -s /usr/local/mpc-1.1.0 /usr/local/mpc
```

8. 设置 LD_LIBRARY_PATH

在编译 GCC 之前，如果不设置 LD_LIBRARY_PATH（如果编译 gmp 时没有指定“--prefix”时安装，一般不用再显示设置），则可能编译时报“error while loading shared libraries: libmpfr.so.6: cannot open shared object file: No such file or directory”等错误。

```
export LD_LIBRARY_PATH=/usr/local/gmp/lib:/usr/local/mpfr/lib:/usr/local/mpc/lib:$LD_LIBRARY_PATH
```

注意：安装 gcc 前可能还要安装（也许也不用，dsx 就没安装 isl 库，之前出错是因为没刷新 terminal）

binutils 库　https://mirrors.tuna.tsinghua.edu.cn/gnu/binutils/

isl 库　ftp://gcc.gnu.org/pub/gcc/infrastructure

9. 编译安装 gcc

在执行 make 编译 GCC 时，有些费时，请耐心等待。在一台 Intel Xeon 2.30GHz 的 48 核 128GB 内存机器上花费 228 分钟（将近 4 个小时，同台机器编译 9.1.0 花费 190 分钟），编译 GCC-8.3.0 的 GCC 版本为 4.8.5（64 位）。

```
tar xzf gcc-8.3.0.tar.gz
```

```
cd gcc-8.3.0

./configure --prefix=/usr/local/gcc-8.3.0 --with-mpfr=/usr/local/mpfr --with-gmp=/usr/local/gmp --with-mpc=/usr/local/mpc

date;time make;date

make install

ln -s /usr/local/gcc-8.3.0 /usr/local/gcc

export PATH=/usr/local/gcc/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/gcc/lib64:$LD_LIBRARY_PATH

export MANPATH=/usr/local/gcc/share/man:$MANPATH

gcc --version
```

在执行 configure 时，如果遇到错误“I suspect your system does not have 32-bit development libraries (libc and headers). If you have them, rerun configure with --enable-multilib. If you do not have them, and want to build a 64-bit-only compiler, rerun configure with --disable-multilib”，表示系统不支持 32 位程序，这样在执行 configure 时需为它支持参数“--disable-multilib”，如：

```
./configure --prefix=/usr/local/gcc-8.3.0 --with-gmp=/usr/local/gmp --with-mpfr=/usr/local/mpfr --with-mpc=/usr/local/mpc --disable-multilib
```

如果 make 时遇到错误“internal compiler error”，可能是因为内存不足，请换台内存更大的机器，或者更换 GCC 版本试试。

如果遇到错误“C compiler cannot create executables”、“error while loading shared libraries: libmpfr.so.6: cannot open shared object file: No such file or directory”或“cannot compute suffix of object files”，可尝试设置 LD_LIBRARY_PATH 后再试试：

```
export LD_LIBRARY_PATH=/usr/local/gmp/lib:/usr/local/mpfr/lib:/usr/local/mpc/lib:$LD_LIBRARY_PATH
```

make 成功结束时的输出：

```
make[8]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/32/libatomic”
make[7]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/32/libatomic”
make[6]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/32/libatomic”
make[5]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/libatomic”
make[4]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/libatomic”
make[3]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/libatomic”
make[2]: 离开目录“/data/GCC/gcc-8.3.0/x86_64-pc-linux-gnu/libatomic”
make[1]: 离开目录“/data/GCC/gcc-8.3.0
```

在完成上列步骤后，检查新安装的 GCC-8.3.0 是否可用：

```
# gcc --version
gcc (GCC) 8.3.0
Copyright © 2018 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。


# man gcc|col -b|grep c++17
       c++17
          GNU dialect of -std=c++17.  The name gnu++1z is deprecated.
       This flag is enabled by default for -std=c++17.
       arguments as an argument for a template template parameter with fewer tem
plate parameters.  This flag is enabled by default for -std=c++17.
       adopted for C++17.  Enabled by default with -std=c++17.  -fstrong-eval-order=s
ome enables just the ordering of member access and shift
       expressions, and is the default without -std=c++17.
       -Wc++17-compat.
       "register" keyword as storage class specifier has been deprecated in C++11 and
 removed in C++17.  Enabled by default with -std=c++17.
     -Wc++17-compat (C++ and Objective-C++ only)
```

已经支持 C++20 标准，但因为标准还未正式发布，所以正式发布后大概率发生变化：

```
# man gcc|col -b|grep c++2a
       c++2a
           GNU dialect of -std=c++2a.  Support is highly experimental, and will almost certainly change in incompatible ways in future releases （支持是高度实验性的，并且在将来的版本中几乎肯定会以不兼容的方式发生变化）.
```

GCC 安装完之后需要在.bashrc 中扩充环境变量 PATH、LD_LIBRARY_PATH、C_INCLUDE_PATH 和 CPLUS_INCLUDE_PATH，否则虽然可以编译程序，但编译后的程序运行时会提示无法找到一些动态库。另外，在 GCC-7.4.0 中编译符合 c++11 的程序时已经不需要显式地给出-std=c++11 的编译选项了。

.bashrc 的备份文件可以到/export/home/lyz/Documents/BACKUP/下找到

10. 编译安装 m4（EDA 实验室有，不用安装）

只有 m4 不可用或版本过低时才需安装 m4（一般路径为/usr/bin/m4），如果 configure 时不指定 prefix，则可省去 export 和 ln 两步：

```
tar xzf m4-1.4.16
cd m4-1.4.16
./configure --prefix=/usr/local/m4-1.4.16
make
make install
ln -s /usr/local/m4-1.4.16 /usr/local/m4
export PATH=/usr/local/m4/bin:$PATH
```

附 1：cmake 支持

在使用 cmake 前，需设置好下列所环境变量，否则 cmake 仍将使用默认目录下的 gcc 和 g++，在 CMakeFiles.txt 文件中设置 CMAKE_C_COMPILER 和 CMAKE_CXX_COMPILER 不能解决这个

问题。如果在此之前已经执行过 cmake，则得先删除 CMakeFiles 目录和文件 CMakeCache.txt，然后再重新执行 cmake 生成 Makefile 文件。

```
export CC=/usr/local/gcc/bin/gcc

export CXX=/usr/local/gcc/bin/g++

export PATH=/usr/local/gcc/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/gcc/lib64:$LD_LIBRARY_PATH

export LD_LIBRARY_PATH=/usr/local/gmp/lib:/usr/local/mpfr/lib:/usr/local/mpc/lib:
$LD_LIBRARY_PATH
```