osx 下搭建操作系统开发环境之 64 位交叉开发工具集（gcc +gdb）v1.0

boxcounter

November 8, 2013

目录

版本记录

- v1.0 – 2013-11-08, 初始发布。

1 前言

《osx 下搭建操作系统开发环境之 32 位交叉开发工具集（gcc+gdb)》一文中讲解了 32 位交叉工具的构建。64 位（专指 x64，非 IA64）的构建方法和 32 位基本相同。看过前文的朋友看本文会发现绝大多数内容都是一样的。没错，之所以不基于前文、只讲差异，是为了独立性，方便只关注 x64 构建方法的朋友。

我的环境：osx 10.9

2 安装 osx 版的 gcc

```
brew install gcc48
```

推荐下载最新的稳定版 gcc。

3 配置编译环境

1. 下载 gcc 源码

   根据参考资料 1 的建议，最好使用最新的 gcc 来进行编译，被编译的源码也推荐使用一样版本的。也就是说，用 gcc 4.8.2 来编译 gcc 4.8.2 的源码。

   下载源码包并解压，得到的目录名称之为“$gcc-4.8.2”。

2. 下载 gcc 依赖项

   需要的依赖项有：

   (a) GNU Binutils

   (b) GMP

   (c) MPFR

   (d) MPC

   将它们都解压出来，把解压出来的 (b)、(c)、(d) 的目录都放到 gcc 源码目录下。都需要去掉版本号，比如解压出来的目录名为“mpc-1.0.1”，那么现在就是“$gcc-4.8.2/mpc”。(a) 无需这么做，因为它需要单独编译，参考后续的步骤 4。

   其中 GMP 源码包是 lzip 压缩格式，需要下载 lzip 工具解压（brew 安装)。

3. 下载 gdb 源码

   下载源码包并解压，得到的目录名称之为“$gdb-7.6.1”。

4. 设置环境变量

```
export CC=/usr/local/bin/gcc-4.8
export CXX=/usr/local/bin/g++-4.8
export CPP=/usr/local/bin/cpp-4.8
export LD=/usr/local/bin/gcc-4.8
```

   这些都是 brew 版 gcc4.8.2 的软链接。如果不设置，那么会使用系统中默认自带的工具，这些工具的版本可能比较老。

```
export PREFIX=$HOME/opt/cross
export TARGET=x86_64-pc-linux-gnu
export PATH="$PREFIX/bin:$PATH"
```

   这些是编译时候使用的选项。需要注意的是：osdev 上的《GCC Cross-Compiler for x86 64》建议将 TARGET 宏定义为 x86_64-elf，但是我实验发现这样行不通，编译 gdb 的时候会报错“configure: error: configuration x86_64-pc-elf is unsupported.”。

5. 编译交叉版的 binutils

```
cd $binutils-x.y.z
mkdir build-binutils
cd build-binutils
../configure --target=$TARGET --prefix=$PREFIX --enable-64-bit-bfd --enable-werror=no
make
make install
```

4 编译交叉版的 gcc

```
cd $gcc-4.8.2
mkdir build-gcc
cd build-gcc
../configure --target=$TARGET --prefix="$PREFIX" --disable-nls --enable-languages=c,c++ --
without-headers
make all-gcc
make all-target-libgcc
make install-gcc
make install-target-libgcc
```

完成后，在“~/opt/cross/bin”下就能看到编译好的交叉版的编译套件了，包括“x86_64-pc-linux-gnu-gcc”、“x86_64-pc-linux-gnu-g++”和“x86_64-pc-linux-gnu-ld ”等等。可以用“$HOME/opt/cross/bin/$TARGET-gcc -version”来验证一下版本是否正确。

另外，为了方便使用，可以在.bashrc 或者.zshrc 中调整环境变量:

```
export PATH="$HOME/opt/cross/bin:$PATH"
```

5 编译交叉版的 gdb

```
cd $gdb-7.6.1
mkdir build-$TARGET
cd build-$TARGET
../configure --target=$TARGET --prefix="$PREFIX" --disable-nls --enable-64-bit-bfd --enable-
werror=no CFLAGS="-m64"
make
sudo make install
```

完成后，在“~/opt/cross/bin”下就能看到编译好的交叉版的 x86_64-pc-linux-gnu-gdb 了。注:

1. 和 x64 的 bochs 配合调试的时候，需要切换到 x86-64 模式（默认模式是 i386）:

   ```
   (gdb) set architecture i386:x86-64
   The target architecture is assumed to be i386:x86-64
   ```

2. 在这个编译选项版本之前，我尝试了好几种其他的编译选项，虽然都能编译出 gdb，但是都无法与 x64 的 bochs 进行配合调试。包括使用“../configure –enable-targets=all –enable-64-bit-bfd”。无法与 x64 的 bochs 进行配合调试的现象是，gdb 能够下断，但是中断点都是错误的，比如:

   ```
   (gdb) target remote localhost:1234
   Remote debugging using 192.168.1.16:1234
   0x00000000 in ?? ()
   (gdb) b *0x7c00
   Breakpoint 1 at 0x7c00
   (gdb) c
   Continuing.

   Program received signal SIGTRAP, Trace/breakpoint trap.
   0x000e0000 in ?? ()
   ```

   正常应该是这样:

   ```
   (gdb) target remote 192.168.1.16:1234
   Remote debugging using localhost:1234
   0x000000000000fff0 in ?? ()
   (gdb)
   ```

总结，gdb 的 TARGET 要和 bochs（而非被调试 OS）的平台一致，即如果使用的是 x64 的 bochs，那么 gdb 的编译 TARGET 也需要是 x64 的。

6 参考资料

1. GCC Cross-Compiler
2. GCC Cross-Compiler for x86 64
3. gdb-remote
4. bochs + GDB help