php-gd 扩展库交叉编译

runsisi@hust.edu.cn @2012/10/05

1. 编译环境

OS: LinuxDeepin 12.06 (based on ubuntu 12.04) 32bit（以下的讨论亦只限于 GNU/Linux 环境下）

HW: Intel Pentium processor T4300, DDRII 667 3GB

编译器：arm-linux-gnueabi-gcc，理论上其他 powerpc，mingw 等 GCC 均可以

Src version:

Php: v5.4.7 zlib: v1.2.7 libpng: v1.5.12 libjpeg: v8d

2. 交叉编译和本地编译的区别

1）输入

相同，同样是那些源代码，配置脚本

2）输出

不同，交叉编译的输出一般为在其他 cpu 架构/系统上（更确切的说是具有不同 ABI 接口的系统上）运行或加载的可执行文件或库，它的输出一般情况下不能在交叉编译器运行的平台下运行或被加载

3）工具

不能简单的说相同或者不同，对于编译链接等用到的工具链肯定不同，例如在我的机器上装了四个针对不同目标平台的 gcc 编译器：

①runsisi@runsisi-Aspire-4736Z:~/Desktop$ gcc -v

Using built-in specs.

COLLECT_GCC=gcc

COLLECT_LTO_WRAPPER=/usr/lib/gcc/i686-linux-gnu/4.6/lto-wrapper

Target: i686-linux-gnu

②runsisi@runsisi-Aspire-4736Z:~/Desktop$ arm-linux-gnueabi-gcc -v

Using built-in specs.

COLLECT_GCC=arm-linux-gnueabi-gcc

COLLECT_LTO_WRAPPER=/usr/lib/gcc/arm-linux-gnueabi/4.6/lto-wrapper

Target: arm-linux-gnueabi

③COLLECT_GCC=/opt/eldk-5.2.1/powerpc/sysroots/i686-eldk-linux/usr/bin/powerpc-linux/powerpc-linux-gcc

COLLECT_LTO_WRAPPER=/opt/eldk-5.2.1/powerpc/sysroots/i686-eldk-linux/usr/libexec/powerpc-linux/gcc/powerpc-linux/4.6.4/lto-wrapper

Target: powerpc-linux

④runsisi@runsisi-Aspire-4736Z:~/Desktop$ i686-w64-mingw32-gcc -v

Using built-in specs.

COLLECT_GCC=i686-w64-mingw32-gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/i686-w64-mingw32/4.6/lto-wrapper
Target: i686-w64-mingw32
注意每一个打印的最后一行”Target“，表明该编译器的输出是针对该平台的。
编译用到的工具除了编译工具链之外，还有 configure 脚本等（当然对于不是使用 autotools 进行源代码发布的开源工程，本段就没有讨论的意义了），对于交叉编译而言，configure 脚本需要使用不同的参数进行调用，有两个最重要的参数：1）指定交叉编译器，可以在执行 configure 脚本之前执行 export CC=arm-linux-gnueabi-gcc，或者像这样：./configure CC=arm-linux-gnueabi-gcc 作为 configure 脚本参数进行指定；2）由于交叉编译出来的可执行文件在当前的平台无法运行，所以另外一个参数是告诉 configure 脚本，我们当前是要进行交叉编译，一些需要通过编译测试程序然后运行测试程序才能得到编译参数的操作就不要进行了，当然还有其他一些编译参数可能针对不同的平台有默认值，该指定交叉编译目标平台的参数以 configure 脚本参数的形式进行指定，如：./configure –-host=arm-linux，具体—host 之后指定的目标平台是 arm-linux, powerpc-linux 还是别的貌似不是太重要，只要不是当前编译器所在平台可以运行的平台就可以了，但是我们还是针对自己需要的目标平台指定为好
4）gcc 默认搜索路径
gcc 编译器在编译和调用 ld 进行链接的时候会去默认路径下搜索头文件和库文件，可以使用 gcc 的 -print-search-dirs 参数打印当前的默认搜索路径，如：
runsisi@runsisi-Aspire-4736Z:~/Desktop$ i686-w64-mingw32-gcc -print-search-dirs
install: /usr/lib/gcc/i686-w64-mingw32/4.6/
programs: =/usr/lib/gcc/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/:/usr/lib/gcc/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/:/usr/lib/gcc/i686-w64-mingw32/4.6/../../../../i686-w64-mingw32/bin/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/4.6/../../../../i686-w64-mingw32/bin/
libraries: =/usr/lib/gcc/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/4.6/../../../../i686-w64-mingw32/lib/i686-w64-mingw32/4.6/:/usr/lib/gcc/i686-w64-mingw32/4.6/../../../../i686-w64-mingw32/lib/../lib/:/usr/lib/gcc/i686-w64-mingw32/4.6/../../../../i686-w64-mingw32/lib/
从打印可以看出，交叉编译器并不会从当前系统目录下搜索头文件和库，所以不需要担心在交叉编译器会搞混。注意：如果在 CFLAGS 中使用—sysroot 指定 root 的位置的话，用户自定义的-I，-L 都会以--sysroot 指定的 root 为 root

3. gd 扩展库编译方式
对于编译 gd 扩展库，存在两种方法，下面分别说明：
1）直接和 php 集成，编译进最终的 php 可执行文件中，在 php v4.x.x 版本之后 php 的源代码里面就带了 gd 扩展库的源代码（在源代码树 ext/gd 下面）（具体版本为多少不关心，我也不是搞 php 的，反正 v5.x.x 肯定是这样的 ~ ），在执行./configure 编译 php 时带上—with-gd 的选项就表明将 gd 编译进 php，不过 php 官方说 gd 库还依赖 libpng 和 libjpeg，而

libpng 又依赖 libz，所以在./configure 的选项中必须同时指定这三个库的位置，以下为我的编译选项：

runsisi@runsisi-Aspire-4736Z:~/Desktop/php/php-5.4.7$ export CC=arm-linux-gnueabi-gcc

runsisi@runsisi-Aspire-4736Z:~/Desktop/php/php-5.4.7$./configure --prefix=/home/runsisi/target --with-gd --disable-libxml --disable-dom --with-zlib-dir=/home/runsisi/target --with-jpeg-dir=/home/runsisi/target --with-png-dir=/home/runsisi/target --host=arm-linux --disable-simplexml --disable-xml --disable-xmlreader --disable-xmlwriter --without-pear –disable-phar

注意五个用特殊颜色标记出来了的部分，紫色部分指定交叉编译器；粉红色—prefix 指定 make install 时 php 被安装到的位置，这个非常重要，如果不指定，会安装到系统默认的/usr/local 下面，而我们交叉编译的 php 在当前系统是根本无法执行的，这会把系统的 php 搞乱，所以一定要指定自己的安装路径；绿色的—with-gd 表示我们要集成 gd 库；深红色的选项指定 zlib，jpeg，png 三个库的位置，注意这三个库也必须是使用交叉编译器编译出来的库；蓝色的—host 表明我们现在在进行交叉编译；其他选项不要问我，我只是为了在编译时不出现找不到相应库的错误所以屏蔽掉了那些扩展，具体这些扩展是做什么我也不懂～，最后一个–disable-phar 选项简单说下，如果不屏蔽掉，在 make 最后阶段会出错，解决方法是使用本机的 php 去打包（具体打包什么我也不知道，makefile 里面用到了 phar 这个扩展功能），因为交叉编译的 php 不可能运行，所以肯定会出错，phar 应该就是个和 jar，rar 类似的功能，屏蔽掉应该也没什么问题。

2）既然 gd 库只是个扩展，那么肯定可以单独编译，这应该就是网上所说的什么追加方式进行编译。gd 的官网都已经挂了，那么肯定直接使用 php 维护的 gd 进行编译。

编译的步骤如下：①切换到 php 源代码目录 ext/gd 下面，执行 php 安装目录下的 phpize ②执行 configure，make 一系列操作，在 ext/gd/modules 目录下就有 gd.so 存在了，如果在 configure 时指定 prefix 为 php 的安装目录，那么 make install 就会把 gd 的头文件和 gd.so 都拷贝到 php 安装目录下去.

所有人都说在编译之前需要执行 phpize 这个脚本，但是从来没人说为什么，其实这个脚本做的就是得到一些宏定义，为扩展库生成 configure 脚本等，注意要打开这个文件，对最开始两行的 prefix 和 datarootdir 根据当前 php 实际所在的位置进行修改，比如我当初是把 php make install 安装在/home/runsisi/target 下面，但是后来我把它移到/home/runsisi/Desktop/php-target 下面了，那就要做下面相应的修改：

orig：

prefix='/home/runsisi/target'

datarootdir='/home/runsisi/target/php'

modified：

prefix='/home/runsisi/Desktop/php-target'

datarootdir='/home/runsisi/Desktop/php-target/php'

4. 几点讨论

1）使用直接集成 gd 库的方法时，zlib，png，jpeg 库可以是静态库，也可以是动态库，但

单独编译 gd 库时 zlib，png，jpeg 必须全部是动态库，因为此时 gd 被编译成动态库，具体原因我这点 linker&loader 的知识还不够解释:)，但拿 windows 下生成库的种类来看，可以知道具体链接什么库是很很有讲究的。。。
windows 下 vc 生成库时，一般有如下几种选择：
静态库/动态库 Ⓧ debug 版/release 版 Ⓧ 静态链接/动态链接 C 运行时库
2）在 php 的 Makefile 中有一处宏定义 CFLAGS_CLEAN 硬编码成了-I/usr/include -g -O2 -fvisibility=hidden，这个地方可能需要修改，其实把-I/usr/include 这句去掉都无所谓；在 EXTRA_LDFLAGS_PROGRAM 后面要加上-ldl，不然会链接出错
3）查看交叉编译好的程序依赖什么库，由于没有现成的 ldd，可以使用其他工具，如：
runsisi@runsisi-Aspire-4736Z:~/Desktop/target/bin$ arm-linux-gnueabi-readelf -d php | grep NEEDED
 0x00000001 (NEEDED) Shared library: [libdl.so.2]
 0x00000001 (NEEDED) Shared library: [libm.so.6]
 0x00000001 (NEEDED) Shared library: [libgcc_s.so.1]
 0x00000001 (NEEDED) Shared library: [libc.so.6]
 0x00000001 (NEEDED) Shared library: [ld-linux.so.3]
4）静态库的链接顺序很重要，如果 liba 依赖 libb，请以这样指定链接顺序：-la -lb
5）如果静态库的链接存在循环依赖问题，请使用 ld 的--start-group archives –end-group 选项

5. 具体的编译步骤，分上述的两种方式进行
1）集成进 php 的方式（默认都动态链接 zlib，png，jpeg 等库，如果目录下同时存在静态库和动态库，gcc 会默认选择静态库进行链接）（假设 php，libz，libpng，libjpeg 的源代码压缩包都在同一个目录下）
首先设置交叉编译器（以 arm gcc 为例）
$ export CC=arm-linux-gnueabi-gcc
① 编译 zlib
$ tar xvzf zlib-1.2.7.tar.gz
$ cd zlib-1.2.7/
$ ./configure –prefix=/home/runsisi/target
出现错误：
./ztest8154: 1: ./ztest8154: Syntax error: word unexpected (expecting ")")
Looking for a four-byte integer type... Not found.
由于 zlib 并非使用的标准 autotools，没有考虑交叉编译的情况，这个错误是执行测试程序导致的，用于检测 32bit 整型是 int 还是 long 还是别的，该错误无关紧要，忽略即可
$ make
$ make install
② 编译 libpng
$ cd ..
$ tar xvzf libpng-1.5.12.tar.gz
$ cd libpng-1.5.12/

注意使用 CFLAGS 指定 zlib 的位置
$ ./configure --prefix=/home/runsisi/target CFLAGS="-I/home/runsisi/target/include -L/home/runsisi/target/lib" –host=arm-linux
$ make
$ make install
③ 编译 libjpeg
$ cd ..
$ tar xvzf jpegsrc.v8d.tar.gz
$ cd jpeg-8d/
$ ./configure --prefix=/home/runsisi/target –host=arm-linux
$ make
$ make install
④ 编译 php
$ cd ..
$ tar xvf php-5.4.7.tar
$ ./configure --prefix=/home/runsisi/target --with-gd --disable-libxml --disable-dom --with-zlib-dir=/home/runsisi/target --with-jpeg-dir=/home/runsisi/target --with-png-dir=/home/runsisi/target --host=arm-linux --disable-simplexml --disable-xml --disable-xmlreader --disable-xmlwriter --without-pear –disable-phar
打开 php-5.4.7/Makefile
修改 67 行附近 CFLAGS_CLEAN = -I/usr/include -g -O2 -fvisibility=hidden 为：
CFLAGS_CLEAN = -g -O2 -fvisibility=hidden
修改 76 行附近 EXTRA_LDFLAGS_PROGRAM = -L/home/runsisi/target/lib 为：
EXTRA_LDFLAGS_PROGRAM = -L/home/runsisi/target/lib -ldl
$ make
$ make install
⑤ 将/home/runsisi/target 目录下 php 及依赖的动态库 libz，libpng，libjpeg 等打包好即可
查看 php 依赖的动态库如下：
runsisi@runsisi-Aspire-4736Z:~/target/bin$ arm-linux-gnueabi-readelf -d php | grep NEEDED
 0x00000001 (NEEDED) Shared library: [libdl.so.2]
 0x00000001 (NEEDED) Shared library: [libpng15.so.15]
 0x00000001 (NEEDED) Shared library: [libz.so.1]
 0x00000001 (NEEDED) Shared library: [libjpeg.so.8]
 0x00000001 (NEEDED) Shared library: [libm.so.6]
 0x00000001 (NEEDED) Shared library: [libgcc_s.so.1]
 0x00000001 (NEEDED) Shared library: [libc.so.6]
 0x00000001 (NEEDED) Shared library: [ld-linux.so.3]
整个安装目录（/home/runsisi/target）下的文件如下：

```
.
├── bin
│   ├── cjpeg
│   ├── djpeg
│   ├── jpegtran
│   ├── libpng15-config
│   ├── libpng-config -> libpng15-config
│   ├── php
│   ├── php-cgi
│   ├── php-config
│   ├── phpize
│   ├── rdjpgcom
│   └── wrjpgcom
├── include
│   ├── jconfig.h
│   └── ...
├── lib
│   ├── libjpeg.a
│   ├── libjpeg.la
│   ├── libjpeg.so -> libjpeg.so.8.4.0
│   ├── libjpeg.so.8 -> libjpeg.so.8.4.0
│   ├── libjpeg.so.8.4.0
│   ├── libpng15.a
│   ├── libpng15.la
│   ├── libpng15.so -> libpng15.so.15.12.0
│   ├── libpng15.so.15 -> libpng15.so.15.12.0
│   ├── libpng15.so.15.12.0
│   ├── libpng.a -> libpng15.a
│   ├── libpng.la -> libpng15.la
│   ├── libpng.so -> libpng15.so
│   ├── libz.a
│   ├── libz.so -> libz.so.1.2.7
│   ├── libz.so.1 -> libz.so.1.2.7
│   ├── libz.so.1.2.7
│   ├── php
│   │   └── build
│   │       ├── acinclude.m4
│   │       ├── config.guess
│   │       ├── config.sub
```

```
│   │   ├── libtool.m4
│   │   ├── ltmain.sh
│   │   ├── Makefile.global
│   │   ├── mkdep.awk
│   │   ├── phpize.m4
│   │   ├── run-tests.php
│   │   ├── scan_makefile_in.awk
│   │   └── shtool
│   └── pkgconfig
│       ├── libpng15.pc
│       ├── libpng.pc -> libpng15.pc
│       └── zlib.pc
├── php
│   └── ...
└── share
    └── ...
42 directories, 309 files
```

2）单独编译 gd 扩展库的方式

① 首先设置交叉编译器

②libz，libpng，libjpeg 同样需要按照第一种方式进行交叉编译

③ 假设 libz，libpng，libjpeg 安装在/home/runsisi/libs-target 中，而 php 编译 make install 时的位置在/home/runsisi/target 中，但后来移动至/home/runsisi/Desktop/php-target 中

④ 打开/home/runsisi/Desktop/php-target/bin/phpize，在第 4 行附近，修改：

```
prefix='/home/runsisi/target'
datarootdir='/home/runsisi/target/php'
```

为：

```
prefix='/home/runsisi/Desktop/php-target'
datarootdir='/home/runsisi/Desktop/php-target/php'
```

⑤ 打开/home/runsisi/Desktop/php-target/bin/php-config，

第 4 行附近，修改：

```
prefix="/home/runsisi/target"
datarootdir="/home/runsisi/target/php"
```

为：

```
prefix="/home/runsisi/Desktop/php-target"
datarootdir="/home/runsisi/Desktop/php-target/php"
```

在第 11 行附近，修改：

```
ldflags=" -L/home/runsisi/target/lib"
```

为指向 libz，libpng，libjpeg 的安装位置

ldflags=" -L/home/runsisi/target/libs-target"
在 13 行附近，修改
extension_dir='/home/runsisi/target/lib/php/extensions/no-debug-non-zts-20100525'
为：
extension_dir='/home/runsisi/Desktop/php-target/lib/php/extensions/no-debug-non-zts-20100525'
⑥ 开始编译 gd 扩展库
$ tar xvf php-5.4.7.tar
$ cd php-5.4.7/ext/gd
$ /home/runsisi/Desktop/php-target/bin/phpize
$ ./configure --prefix=/home/runsisi/Desktop/php-target --host=arm-linux --with-php-config=/home/runsisi/Desktop/php-target/bin/php-config --with-zlib-dir=/home/runsisi/libs-target --with-png-dir=/home/runsisi/libs-target --with-jpeg-dir=/home/runsisi/libs-target
$ make
$ make install
查看 gd.so 依赖的动态库如下：
runsisi@runsisi-Aspire-4736Z:~/Desktop/php-target/lib/php/extensions/no-debug-non-zts-20100525$ arm-linux-gnueabi-readelf -d gd.so | grep NEEDED
 0x00000001 (NEEDED)   Shared library: [libpng15.so.15]
 0x00000001 (NEEDED)   Shared library: [libz.so.1]
 0x00000001 (NEEDED)   Shared library: [libjpeg.so.8]
 0x00000001 (NEEDED)   Shared library: [libc.so.6]
 0x00000001 (NEEDED)   Shared library: [ld-linux.so.3]
php-target 下面的文件如下：
```
.
├── bin
│   ├── php
│   ├── php-cgi
│   ├── php-config
│   └── phpize
├── include
│   └── ...
├── lib
│   └── php
│       ├── build
│       │   ├── acinclude.m4
│       │   ├── config.guess
│       │   ├── config.sub
```

```
│       │   ├── libtool.m4
│       │   ├── ltmain.sh
│       │   ├── Makefile.global
│       │   ├── mkdep.awk
│       │   ├── phpize.m4
│       │   ├── run-tests.php
│       │   ├── scan_makefile_in.awk
│       │   └── shtool
│       └── extensions
│           └── no-debug-non-zts-20100525
│               └── gd.so
└── php
    └── ...
37 directories, 263 files
```