Madplay移植到mini2440全过程详解

madplay交叉编译

交叉编译器：arm-linux-gcc 3.4.1

PC环境：Ubuntu 8.10

注意：最好在root权限下执行以下移植，否则在make install时可能权限不够。

一、移植madplay前的准备

madplay的移植需要以下的几个包：libid3tag-0.15.1b.tar.gz、libmad-0.15.1b.tar.gz、madplay-0.15.2b.tar.gz、zlib-1.2.3.tar.gz。其中前面的三个包可以在http://sourceforge.net/project/showfiles.php?group\_id=12349中下载，后一个包可以在http://www.gzip.org/zlib/zlib-1.2.3.tar.gz中找到。

注意：一般移植嵌入式应用软件的步骤是先在PC上配置编译该软件并运行，以了解一下该软件的用途和使用方法等，现在就让我们先在PC上开始吧。

二、建立工作目录，拷贝源代码包

在/opt/madplay目录下建立madplay目录，并以此为工作目录，并在该目录中建立以下子目录，以存放不同的文件：

# cd /opt

# mkdir madplay

# cd madplay

# mkdir tarball src-x86 src-arm target-x86 target-arm

目录说明：

tarball目录用来存放所有的源代码包

src-x86目录用来存放X86版本的所有源代码文件

src-arm目录用来存放ARM版本的所有源代码文件

target-x86目录是X86版本的安装目录

target-arm目录是ARM版本的安装目录

接下来把从网上下载到的源代码包放入tarball目录。

三、PC-Linux（PC版本）

1、解压源代码包

# cd tarball

# for f in $(ls \*.tar.gz); do tar -zxvf $f –C ../src-86; done

或者：

# cd tarball

# tar -zxvf zlib-1.2.3.tar.gz -C ../src-86

# tar -zxvf libid3tag-0.15.1b.tar.gz -C ../src-86

# tar -zxvf libmad-0.15.1b.tar.gz -C ../src-86

# tar -zxvf madplay-0.15.2b.tar.gz -C ../src-86

2、编译madplay及其所依赖的库文件

（1）编译zlib，因为libid3tag依赖于这个库文件

# cd ../src-x86/zlib-1.2.3

# ./configure --prefix=/opt/madplay/target-x86

# make

# make install

编译好之后就可以在上面prefix指定目录下的lib目录下找到libz.a这个库。

（2）编译libid3tag

# cd ../libid3tag-0.15.1b

# ./configure --prefix=/opt/madplay/target-x86 --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-x86/include LDFLAGS=-L/opt/madplay/target-x86/lib

# make

# make install

其中，--disable-shared --enable-static是指定为静态编译，不过这样并不能够进行静态编译。至于为什么，有待于继续研究。

（3）编译libmad

# cd ../libmad-0.15.1b

# ./configure --prefix=/opt/madplay/target-x86 --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-x86/include LDFLAGS=-L/opt/madplay/target-x86/lib

注意：如果此时make，make install就会报错，说是"-fforce-mem"参数不能识别。不知道是什么原因，然后就在网上查了一通，好像说是我机器环境的gcc版本（我的是gcc 4.3.2）过高导致的，所以不能识别其中的"-fforce-mem"参数选项，真是搞不懂，版本高还识别不了。。。

于是乎就打开当前目录下的Makefile文件，去掉里面出现的"-fforce-mem"，然后再make，make install就OK了。

（4）编译madplay

# cd ../madplay-0.15.2b

# ./configure --prefix=/opt/madplay/target-x86 --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-x86/include LDFLAGS=-L/opt/madplay/target-x86/lib

# make

# make install

但是，这样得到的是动态链接。如果在madplay-0.15.2b目录中执行gcc命令，并且在最后加上-static，运行以后就可以得到静态链接的程序，如（5）所示。

（5）静态链接

# gcc -Wall -O2 -fomit-frame-pointer -o madplay madplay.o getopt.o getopt1.o version.o resample.o filter.o tag.o crc.o rgain.o player.o audio.o audio\_aiff.o audio\_cdda.o audio\_hex.o audio\_null.o audio\_raw.o audio\_snd.o audio\_wave.o audio\_oss.o -L/opt/madplay/target-x86/lib /opt/madplay/target-x86/lib/libmad.a /opt/madplay/target-x86/lib/libid3tag.a -lz -lm -static

执行完毕后，动态链接执行文件将被安装在我们指定的目录中：/opt/madplay/target-x86

静态链接执行文件将被安装在我们当前的目录中：/opt/madplay/src-x86/madplay-0.15.2b

# ls /opt/madplay/target-x86/bin

abxtest madplay

其中abxtest是附加生成的测试程序，不必理会。

使用file命令检查一下：

# file /opt/madplay/target-x86/bin/madplay

madplay: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.8, dynamically linked (uses shared libs), not stripped

# file /opt/madplay/src-x86/madplay-0.15.2b

madplay: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.8, statically linked, not stripped

上面的信息显示了我们编译得到的两个链接执行程序：动态链接和静态链接。

# arm-linux-readelf -d /opt/madplay/target-x86/bin/madplay

输出一大堆信息，最上面显示了动态链接编译所需的动态库：libm.so.6和libc.so.6。

静态链接文件大小为862.8KB（883531字节），动态链接文件大小仅仅为279.5KB（286206字节），但是运行时需要某些动态库。

3、测试PC-Linux版本的madplay

拷贝一首mp3文件到madplay所在的目录，执行：

# ./madplay test.mp3

如果你的声卡已经正确安装，就可以听到mp3 的声音了。

4、构建Shell编译脚本build-x86.sh

通过以上步骤，我们看到配置和编译不仅有一定的顺序，还需要注意一些安装细节，虽然这次编译通过了，但不免以后会忘记这个过程，特别是当程序更加复杂的时候，因此我们要养成好习惯，把整个过程构建为一个脚本，以后只要执行这个脚本就可以完成所有步骤了，如下所示是PC-Linux版本的madplay构建脚本，该脚本位于madplay工作目录的根目录下。

build-x86.sh

#!/bin/sh

# This build script is for madplay under PC-Linux

MADPLAY\_DIR=/opt/madplay

TARBALL\_DIR=$MADPLAY\_DIR/tarball

SRC\_DIR=$MADPLAY\_DIR/src-x86

TARGET\_DIR=$MADPLAY\_DIR/target-x86

tar -zxvf $TARBALL\_DIR/zlib-1.2.3.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/libid3tag-0.15.1b.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/libmad-0.15.1b.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/madplay-0.15.2b.tar.gz -C $SRC\_DIR

# for f in $(ls $TARBALL\_DIR/\*.tar.gz); do tar -zxvf $f -C $SRC\_DIR; done

cd $SRC\_DIR/zlib-1.2.3

./configure --prefix=$TARGET\_DIR

make

make install

cd $SRC\_DIR/libid3tag-0.15.1b

./configure --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

make

make install

cd $SRC\_DIR/libmad-0.15.1b

./configure --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

sed 's/-fforce-mem //g' Makefile > Makefile.bak

mv Makefile.bak Makefile

make

make install

cd $SRC\_DIR/madplay-0.15.2b

./configure --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

make

make install

gcc -Wall -O2 -fomit-frame-pointer -o madplay-s madplay.o getopt.o getopt1.o version.o resample.o filter.o tag.o crc.o rgain.o player.o audio.o audio\_aiff.o audio\_cdda.o audio\_hex.o audio\_null.o audio\_raw.o audio\_snd.o audio\_wave.o audio\_oss.o -L$TARGET\_DIR/lib $TARGET\_DIR/lib/libmad.a $TARGET\_DIR/lib/libid3tag.a -lz -lm -static

mv madplay madplay-d

mv $TARGET\_DIR/bin/madplay $TARGET\_DIR/bin/madplay-d

cp -a madplay-s $TARGET\_DIR/bin

四、ARM-Linux（ARM版本）

1、解压源代码包

# cd tarball

# for f in $(ls \*.tar.gz); do tar -zxvf $f –C ../src-arm; done

或者：

# cd tarball

# tar -zxvf zlib-1.2.3.tar.gz -C ../src-arm

# tar -zxvf libid3tag-0.15.1b.tar.gz -C ../src-arm

# tar -zxvf libmad-0.15.1b.tar.gz -C ../src-arm

# tar -zxvf madplay-0.15.2b.tar.gz -C ../src-arm

2、编译madplay及其所依赖的库文件

（1）设定编译器环境变量

# export CC=arm-linux-gcc

（2）编译zlib，因为libid3tag依赖于这个库文件

# cd ../src-arm/zlib-1.2.3

# ./configure --prefix=/opt/madplay/target-arm

# make

# make install

或者：

打开/opt/madplay/src-arm/zlib-1.2.3

目录下的Makefile文件，按以下内容修改并保存：

CC=arm-linux-gcc

AR=arm-linux-ar rcs

RANLIB=arm-linux-ranlib

prefix=/opt/madplay/target-arm

然后make，make install，编译好之后就可以在上面prefix指定目录下的lib目录下找到libz.a这个库。

（3）编译libid3tag

# cd ../libid3tag-0.15.1b

# ./configure --host=arm-linux --prefix=/opt/madplay/target-arm --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-arm/include LDFLAGS=-L/opt/madplay/target-arm/lib

# make

# make install

（4）编译libmad

# cd ../libmad-0.15.1b

# ./configure --host=arm-linux --prefix=/opt/madplay/target-arm --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-arm/include LDFLAGS=-L/opt/madplay/target-arm/lib

# make

# make install

（5）编译madplay

# cd ../madplay-0.15.2b

# ./configure --host=arm-linux --prefix=/opt/madplay/target-arm --disable-debugging --disable-shared --enable-static CPPFLAGS=-I/opt/madplay/target-arm/include LDFLAGS=-L/opt/madplay/target-arm/lib

# make

# make install

但是，这样得到的是动态链接。如果在madplay-0.15.2b目录中执行arm-inux-gcc命令，并且在最后加上-static，运行以后就可以得到静态链接的程序，如（6）所示。

（6）静态链接

# arm-linux-gcc -Wall -O2 -fomit-frame-pointer -o madplay-s madplay.o getopt.o getopt1.o version.o resample.o filter.o tag.o crc.o rgain.o player.o audio.o audio\_aiff.o audio\_cdda.o audio\_hex.o audio\_null.o audio\_raw.o audio\_snd.o audio\_wave.o audio\_oss.o -L/opt/madplay/target-arm/lib /opt/madplay/target-arm/lib/libmad.a /opt/madplay/target-arm/lib/libid3tag.a -lz -lm -static

执行完毕后，动态链接可执行文件将被安装在我们指定的目录中：/opt/madplay/target-arm

静态链接可执行文件将被安装在我们当前的目录中：/opt/madplay/src-arm/madplay-0.15.2b

# ls /opt/madplay/target-arm/bin

abxtest madplay

其中abxtest是附加生成的测试程序，不必理会。

使用file命令检查一下：

# file /opt/madplay/target-arm/bin/madplay

madplay: ELF 32-bit LSB executable, ARM, version 1, for GNU/Linux 2.4.3, dynamically linked (uses shared libs), not stripped

# file /opt/madplay/src-arm/madplay-0.15.2b

madplay: ELF 32-bit LSB executable, ARM, version 1, for GNU/Linux 2.4.3, statically linked, not stripped

上面显示了我们编译得到的两个链接可执行程序：动态链接和静态链接。

# arm-linux-readelf -d /opt/madplay/target-arm/bin/madplay

最上面显示了动态链接编译所需的动态库：libm.so.6和libc.so.6。

静态链接文件大小为922.2KB（944377字节），动态链接文件大小仅仅为285.8KB（292690字节），但是运行时需要某些动态库。

3、下载madplay到开发板运行测试

为了区别于板子中已经存在的madplay，我们把新做的改名为mymadplay，把它以及依赖库通过FTP或者NFS方式下载到开发板，并作如下放置：

执行文件mymadplay放在/usr/bin/目录，库文件：libm.so.6和libc.so.6放在/usr/lib目录，当然如果是静态链接的可执行程序就不用拷贝库文件了。

拷贝一首mp3文件到开发板上，然后执行mymadplay进行测试。

4、构建并修正ARM版本的编译脚本build-arm.sh

既然我们已经构建了一个简单易用的编译脚本，现在就可以通过对它稍作修改来进行交叉编译了，这就是通常所说的移植。简单的移植只要重新指定一下编译器就可以了，可以通过修改环境变量来实现。很多的移植所要修改的环境变量是：

CC：编译器，系统默认为gcc

AR：库工具，用以创建和修改库，系统默认ar

LD：链接器，系统默认为LD

RANLIB：随机库创建器，系统默认为ranlib

AS：汇编器，系统默认为as

NM：库查看工具，系统默认为nm

还有一些不常用的其他环境变量，在此就不一一列举了。

需要注意的是，并不是每个移植都需要做全面的环境变量修改，有些是不需要改的，这要根据实际情况，也就是系统提示信息来调整。

除了要修改编译器环境变量，一般还需要在配置中加入目标平台指定标识，在此为"arm-linux"，修改后的脚本如下：

build-arm.sh

#!/bin/sh

# This build script is for madplay under ARM-Linux

MADPLAY\_DIR=/opt/madplay

TARBALL\_DIR=$MADPLAY\_DIR/tarball

SRC\_DIR=$MADPLAY\_DIR/src-arm

TARGET\_DIR=$MADPLAY\_DIR/target-arm

tar -zxvf $TARBALL\_DIR/zlib-1.2.3.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/libid3tag-0.15.1b.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/libmad-0.15.1b.tar.gz -C $SRC\_DIR

tar -zxvf $TARBALL\_DIR/madplay-0.15.2b.tar.gz -C $SRC\_DIR

# for f in $(ls $TARBALL\_DIR/\*.tar.gz); do tar -zxvf $f -C $SRC\_DIR; done

export CC=arm-linux-gcc

cd $SRC\_DIR/zlib-1.2.3

./configure --prefix=$TARGET\_DIR

make

make install

cd $SRC\_DIR/libid3tag-0.15.1b

./configure --host=arm-linux --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

make

make install

cd $SRC\_DIR/libmad-0.15.1b

./configure --host=arm-linux --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

make

make install

cd $SRC\_DIR/madplay-0.15.2b

./configure --host=arm-linux --prefix=$TARGET\_DIR --disable-debugging --disable-shared --enable-static CPPFLAGS=-I$TARGET\_DIR/include LDFLAGS=-L$TARGET\_DIR/lib

make

make install

arm-linux-gcc -Wall -O2 -fomit-frame-pointer -o madplay-s madplay.o getopt.o getopt1.o version.o resample.o filter.o tag.o crc.o rgain.o player.o audio.o audio\_aiff.o audio\_cdda.o audio\_hex.o audio\_null.o audio\_raw.o audio\_snd.o audio\_wave.o audio\_oss.o -L$TARGET\_DIR/lib $TARGET\_DIR/lib/libmad.a $TARGET\_DIR/lib/libid3tag.a -lz -lm -static

mv madplay madplay-d

mv $TARGET\_DIR/bin/madplay $TARGET\_DIR/bin/madplay-d

cp -a madplay-s $TARGET\_DIR/bin

PS：

在PC机和开发板上执行madplay程序的过程信息：

#madplay XXX.mp3

MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-2004 Robert Leslie et al.

Title: XXXXXX

Artist: XXXXXX

Year: XXXXXX

Genre: XXXXXX

出现这些信息就成功了，音质还真不错，再加个图形界面和按键，就能做个MP3了！

Madplay还有一些参数：比如-r循环播放，还可以进行格式转换：

madplay 111.mp3 –o 222.wav。

功能很强大。