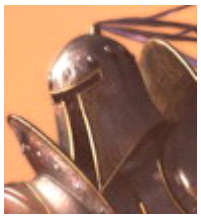
MiniGUI-1.6.10

在 mini2440 上的移植



--------By LastRitter

QQ:314665354

E-mail:superyongzhe@163.com

2009 年 8 月 3 日

一、 编译前准备

1. 硬件环境

A. 主机

x86 系列 PC 机。

B. 开发板

友善之臂 mini2440 开发板，VGA 扩展板和 CRT 显示器。

C. 通讯连接

串口和 USB。

2. 软件环境

A. 主机操作系统

VMware 虚拟机下的 Redhat linux 9.0(完全安装) 。

B. 开发板操作系统

嵌入式 Linux，内核版本为 2.6.13。

C. 交叉编译器

arm-linux-gcc-3.3.2（安装位置：/usr/local/arm/3.3.2）。

D. 通讯方式

虚拟机与主机（Windows XP）使用共享文件夹通讯，主机与开发板使用 DNW 和 Secure CRT 通讯。

3. 相关说明

A. 命令

所有以#开头的均为在 shell 中执行的命令。

B. 交叉编译器位置

交叉编译器的安装位置为/usr/local/arm/3.3.2，进行交叉编译产生的库都安装在/usr/local/arm/3.3.2/arm-linux 目录下，您可以根据你的交叉编译器的实际位置进行更改。

C. 工作目录

默认的工作目录为/root/minigui，绝大多数操作都在此目录下进行。

D. 注释

对 shell 脚本进行的注释均放在该命令下一行的圆括号内。

4. 下载 MiniGUI 图形库源码

地址：http://www.minigui.org/downloads

A. 开发库 **libminigui-1.6.10.tar.gz**

编译完成后生成三个库：它们分别是 libminigui、libmgext 以及 libvcongui。

libminigui 是提供窗口管理和图形接口的核心函数库，也提供了大量的标准控件；

libmgext 是 libminigui 的一个扩展库，提供了一些高级控件以及“文件打开”、“颜色选择”对话框等；

libvcongui 则为 Linux 操作系统提供了一个应用程序可用的虚拟控制台窗口，从而可以方便地在 MiniGUI 环境中运行字符界面的应用程序。

B. 资源文件 **minigui-res-1.6.10.tar.gz**

MiniGUI 所使用的资源， 包括基本字体、图标、位图和鼠标光标。

C. 示例程序 **mg-samples-str-1.6.10.tar.gz**

《MiniGUI 编程指南》的配套示例程序。

D. 综合演示程序 **mde-1.6.10.tar.gz**

MiniGUI 的综合演示程序包， 其中包含有一些较为复杂的示例程序。

5. 下载所依赖函数库源码

A. zlib 库

这是编译后面的库的基础。地址：http://www.zlib.net/zlib-1.2.3.tar.gz

B. png 库

png 图片支持库。地址：http://superb-east.dl.sourceforge.net/sourceforge/libpng/libpng-1.0.10rc1.tar.gz

C. jpeg 库

jpeg 图片支持库。地址：

http://www.minigui.org/downloads

D. ttf 库

TrueType 字体的支持库。地址：

http://nchc.dl.sourceforge.net/sourceforge/freetype/freetype-1.3.1.tar.gz

E. popt 库

编译 mde 需要。地址：

http://gd.tuwien.ac.at/utils/rpm.org/dist/rpm-4.1.x/popt-1.7.tar.gz

6. 下载上位机仿真程序

Qt 虚拟缓冲帧 qvfb-1.1.tar.gz，上位机仿真需要。

7. 复制根文件系统

复制 mini2440 开发板自带的根文件系统 root_default.tgz 到虚拟机中，并把 yaffs 映像制作工具 mkyaffsimage 复制到 /bin 目录下。

8. 解压源码包

A. 建立工作目录及复制源码包

#mkdir /root/minigui

(创建工作目录，绝大部分相关的操作抖将在此目录下进行)

#cd /root/minigui

(进入工作目录)

#mkdir source

(创建 source 目录，所有的源码包都放在此目录)

#cp … source

(把所有下载的源码包复制到此目录)

#mkdir arm

(创建 arm 目录，所有进行交叉编译所需的源码都放在此目录)

#mkdir x86

（创建 x86 目录，所有进行仿真所需的源码都放在此目录）

B. 解压上位机仿真所需源码

#cd /root/minigui/source

(进入源码包所在目录)

tar zxvf libminigui-1.6.10.tar.gz –C ../x86

(解压 MiniGUI 图形库源码到 x86 目录，-C 选项的含义是更改解压缩目录)

tar zxvf minigui-res-1.6.10.tar.gz –C ../x86

(解压 MiniGUI 资源文件到 x86 目录)

tar zxvf mg-samples-str-1.6.10.tar.gz –C ../x86

(解压 MiniGUI 示例源码到 x86 目录)

tar zxvf mde-1.6.10.tar.gz –C ../x86

(解压 MiniGUI 综合演示源码到 x86 目录)

tar zxvf qvfb-1.1.tar.gz –C ../x86

(解压 qvfb 源码到 x86 目录)

C. 解压交叉编译所需源码

#cd /root/minigui/source

(进入源码包所在目录)

#tar zxvf libminigui-1.6.10.tar.gz –C ../arm

(解压 MiniGUI 图形库源码到 arm 目录)

#tar zxvf minigui-res-1.6.10.tar.gz –C ../arm

(解压 MiniGUI 资源文件到 arm 目录)

#tar zxvf mg-samples-str-1.6.10.tar.gz –C ../arm

(解压 MiniGUI 示例源码到 arm 目录)

#tar zxvf mde-1.6.10.tar.gz –C ../arm

(解压 MiniGUI 综合演示源码到 arm 目录)

#tar zxvf zlib-1.2.3.tar.gz –C ../arm

(解压 zlib 源码到 arm 目录)

#tar zxvf libpng-1.0.10rc1.tar.gz –C ../arm

(解压 png 源码到 arm 目录)

#tar zxvf jpegsrc.v6b.tar.gz –C ../arm

(解压 jpeg 源码到 arm 目录)

#tar zxvf popt-1.7.tar.gz –C ../arm

(解压 popt 源码到 arm 目录)

D. 解压根文件系统

#tar zxvf root_default.tgz –C ../arm

(解压根文件系统 root_default 到 arm 目录)

9. 目录及源码分布总览

/root/minigui

|source

libminigui-1.6.10.tar.gz

minigui-res-1.6.10.tar.gz

mg-samples-str-1.6.10.tar.gz

mde-1.6.10.tar.gz

zlib-1.2.3.tar.gz

libpng-1.0.10rc1.tar.gz

jpegsrc.v6b.tar.gz

freetype-1.3.1.tar.gz

popt-1.7.tar.gz

qvfb-1.1.tar.gz

root_default.tgz

|x86

|libminigui-1.6.10.

|minigui-res-1.6.10

|mg-samples-str-1.6.10

|mde-1.6.10

|qvfb-1.1

|arm

|libminigui-1.6.10.

|minigui-res-1.6.10

|mg-samples-str-1.6.10

|mde-1.6.10

|zlib-1.2.3

|libpng-1.0.10rc1

|jpegsrc.v6b

|freetype-1.3.1

|popt-1.7

|root_default

二、 建立仿真开发环境

1. 安装 minigui-res-1.6.10

cd /root/minigui/x86/minigui-res-1.6.10

make install

这样 MiniGUI 运行时所需的资源文件就被安装到 /usr/local/lib/minigui/res/目录下。

2. 编译 libminigui-1.6.10

#cd /root/minigui/x86/libminigui-1.6.10

#./configure

#make

#make install

库文件被安装到系统中。

3. 编译 mg-samples-str-1.6.10

#cd /root/minigui/x86/mg-samples-str-1.6.10

#./configure

#make

在 src 目录下可以看到生成的可执行程序。

4. 编译 mde-1.6.10

#cd /root/minigui/x86/mde-1.6.10

#./configure

#make

在各个子目录里可以看到相应的可执行演示程序。

5. 编译 qvfb-1.1

#cd /root/minigui/x86/qvfb-1.1

#./configure

#make

在 qvfb 子目录下可以看到可执行的 qvfb 程序，复制到系统目录即可运行。如果在先前按照 mini2440 的用户手册搭建过 Qt/Embedded 开发环境，可能无法编译成功，主要是因为修改了/etc/ld.so.config 文件。最简单的方法就是把搭建 Qt/Embedded 开发环境时生成的 qvfb 复制到/bin 目录，直接就可以运行。这里“编译 qvfb-1.1”这一步就可以省略掉了。

6. 在上位机上仿真 MiniGUI 应用程序

#qvfb –width 640 –height 480 &

(由于前面把 qvfb 复制到了/bin 目录，所以可以直接运行。后面的参数制定了 qvfb 的显示尺寸，因为在 MiniGUI。cfg 里 qvfb 的尺寸默认为 640*480，所以如果不指定这个值会无法运行，&选项指名这个程序在后台执行。)

cd /root/minigui/x86/ mg-samples-str-1.6.10/src

#./helloworld

如果一切顺利的话现在就可以在 qvfb 上面看到 MiniGUI 的界面了。还有 mde 子目录里面的程序，也可以直接执行。

三、 交叉编译图形库

1. 编译 zlib 库

由于 zlib 库的 configure 脚本不支持交叉编译选项，所以我们只好使用符号链接把 gcc 指向我们的交叉编译器 arm-linux-gcc，在编译完后再改回来即可。

A. 把 **gcc** 指向我们的交叉编译器 **arm-linux-gcc**

cd /usr/bin

mv gcc gcc_back

（备份 gcc）

ln -s /usr/local/arm/3.4.1/bin/arm-linux-gcc ./gcc

（创建 gcc 到 arm-linux-gcc 的符号连接）

mv ld ld_back

（备份 ld）

ln -s /usr/local/arm/3.4.1/bin/arm-linux-ld ./ld

（创建 ld 到 arm-linux-ld 的符号连接）

B. 交叉编译 **zlib** 库

#cd /root/minigui/arm/zlib-1.2.3

#./configure --prefix=/usr/local/arm/3.3.2/arm-linux/ --

shared

（prefix 选项把 zlib 库安装在/usr/local/arm/3.3.2/arm-linux/）

（shared 说明生成共享库）

#make

#make install

C. 改回 **gcc**

cd /usr/bin

rm gcc

（删除 gcc 到 arm-linux-gcc 的符号连接）

mv gcc_back gcc

（还原 gcc）

rm ld

（删除 ld 到 arm-linux-ld 的符号连接）

mv ld_back ld

（还原 ld）

2. 编译 png 库

这个是用来显示 png 图形的，MiniGUI 里很多图都是 png 的，如果没有这个库，你的 MiniGUI 将无法正常工作。由于 libpng 不提供有效的 configure 脚本，所以只好自己动

手改写 Makefile 文件了。

A. 改写 **Makefile**

#cd

cp scripts/makefile.linux Makefile

（把 Scripts 下的一个 Makefile 拷出来自己动手改）

vi Makefile

(自己动手改 Makefile)

CC=arm-linux-gcc

prefix=/usr/local/arm/3.3.2/arm-linux

ZLIBLIB=/usr/local/arm/3.3.2/arm-linux/lib

ZLIBINC=/usr/local/arm/3.3.2/arm-linux/include

（保存）

B. 编译安装

make

make install

3. 编译 jpeg 库

由于 jpeg 库的 configure 文件设计的有问题，得先用 gcc 编译一个 dummy.c 的文件，然后才能后面编译，不然的话就会出现 libtool 找不到之类的错误。所以先本机编译，然后清

除后，最后交叉编译即可解决这个问题。

A. 本机编译 **jpeg** 库

cd /root/minigui/arm/jpeg-6b

./configure --enable-shared --enable-static

make

make clean

B. 交叉编译 **jpeg** 库

./configure --prefix=/usr/local/arm/3.3.2/arm-linux/ CC=arm-linux-gcc --enable-shared --enable-static

make

mkdir –p /usr/local/arm/3.3.2/arm-linux/man/man1

(安装前需要在 arm-linux 下建个目录，不然安装会出错)

make install

4. 编译 popt

cd /root/minigui/arm/popt-1.7

./configure --prefix=/usr/local/arm/3.3.2/arm-linux/ --host=arm-linux --enable-shared --enable-static

make

make install

5. 编译 libttf 库

libttf 库是 TrueType 字体的支持库，当然可以支持也可以不支持，这个库只能全手动安装和编译，先建立一个目录来存放其有用的或者说是我们所用到的库的源文件。

A. 复制源文件

#mkdir -p /root/minigui/arm/libttf/extend

#cd /root/minigui/arm/

#cp freetype-1.3.1/lib/* freetype-1.3.1/lib/arch/ansi/* libttf/

#cp freetype-1.3.1/lib/extend/* libttf/extend/

B. 交叉编译

#cd libttf

#arm-linux-gcc -c -fPIC -O2 freetype.c

（这个 C 源文件包括了其它所有的 .c 文件）

#arm-linux-gcc -c -fPIC -O2 -I./ extend/*.c

（把 extend 下所有的 .c 文件全部编译）

#arm-linux-gcc --shared -o libttf.so *.o

（生成最后的动态链接库）

#cp libttf.so /usr/local/arm/3.3.2/arm-linux/lib

6. 编译 libminigui

```
# cd /root/minigui/arm/libminigui-1.6.10
# ./configure
--prefix=/root/minigui/arm/target \
--host=arm-linux \
--target=arm-linux \
--build=i386-linux \
--with-osname=linux \
--with-style=classic \
--with-targetname=fbcon \
--enable-autoial \
--enable-rbf16 \
--disable-vbfsupport \
CC=arm-linux-gcc
# make
# make install
```

（这样就会把交叉编译好的库文件和头文件安装在 /root/minigui/arm/target 目录）

```
#cp /root/minigui/arm/target/lib /*
/usr/local/arm/3.3.2/arm-linux/lib/
```

#cp /root/minigui/arm/target/include /*

/usr/local/arm/3.3.2/arm-linux/include/

（把交叉编译好的库文件和头文件复制到交叉编译器中，便于使用）

7. 编译 mg-samples

A. 修改配置文件

#cd /root/minigui/arm/mg-samples-1.6.10

修改 configure 文件，

在文件的最前面加上交叉编译的工具

CC= arm-linux-gcc

CPP= arm-linux-cpp

LD= arm-linux-ld

AR= arm-linux-ar

RANLIB= arm-linux-ranlib

STRIP= arm-linux-strip

B. 配置

./configure --prefix=/home/nick/minigui/tmp/ --host=arm-linux --target=arm-linux

C. 修改 **Makefile**

CC = arm-linux-gcc

–I/home/nick/minigui/miniguitmp/include

–L/home/nick/minigui/miniguitmp/lib

CFLAGS =-O2

LIBOBJS = -lminigui -lmgext -lm -lpthread

LIBS = -lminigui -lmgext -lm -lpthread

COMPILE = $(CC) $(DEFS) $(DEFAULT_INCLUDES)

$(INCLUDES) $(AM_CPPFLAGS) \$(CPPFLAGS)

$(AM_CFLAGS) $(CFLAGS) -lminigui -lmgext -lm -lpthread

D. 编译

make

make install

8. 编译 mde

与编译 mg-samples 方法相同。

四、 部署图形库

1. 复制 MiniGUI 图形库到根文件系统

#cp /root/minigui/arm/target/lib/*
/root/minigui/arm/root_default/lib

2. 复制依赖库到根文件系统

#cp /usr/local/arm/3.3.2/arm-linux/lib/libjpeg*
/root/minigui/arm/root_default/lib

#cp /usr/local/arm/3.3.2/arm-linux/lib/libm*
/root/minigui/arm/root_default/lib

#cp /usr/local/arm/3.3.2/arm-linux/lib/libpng*
/root/minigui/arm/root_default/lib

#cp /usr/local/arm/3.3.2/arm-linux/lib/libpopt*
/root/minigui/arm/root_default/lib

#cp /usr/local/arm/3.3.2/arm-linux/lib/libttf*
/root/minigui/arm/root_default/lib

#cp /usr/local/arm/3.3.2/arm-linux/lib/libz*
/root/minigui/arm/root_default/lib

3. 缓存函数库

#cd /root/minigui/arm/root_default/etc

#vi ld.so.cfg

（添加）

/usr/local/lib

/usr/lib

/lib

（保存）

#ldconfig –r /root/minigui/arm/root_default

如果不这样做，在开发板上运行 MiniGUI 应用程序时可能会出现找不到库的情况。

4. 复制资源文件

#mkdir –p /root/minigui/arm/root_default/usr/local/lib/minigui

#cp /usr/local/lib/minigui/res /root/minigui/arm/root_default/usr/local/lib/minigui –r

（MiniGUI 应用程序在运行时可能会用到的一些位图，光标等资源文件。）

5. 复制 MiniGUI.cfg 到根文件系统

#cp /root/minigui/arm/target/etc/MiniGUI.cfg /root/minigui/arm/root_default/usr/local/etc

（这是 MiniGUI 运行时配置文件，在 MiniGUI 应用程序运行时会用到。）

6. 修改配置文件

#cd /root/minigui/arm/root_default/usr/local/etc

#vi MiniGUI.cfg

(修改)

[system]

gal_engine=fbcon

（设置图形引擎为帧缓冲控制台 fbcon）

ial_engine=cosole

（设置输入引擎为控制台）

mdev=/dev/input/mice

（设置鼠标输入设备）

mtype=IMSP2

（设置输入法）

[fbcon]

Defaultmode=1024x768-16bpp

（设置 fbcon 显示参数，我使用的是 VGA 板，如果你使用的时 3.5 寸液晶，则 Defaultmode=320x240-16bpp）

（保存）

7. 修改启动脚本

#cd /root/minigui/arm/root_default/etc/init.d

#vi rcS

（在最后添加如下内容）

/bin/ln -s /dev/vc/0 /dev/tty0

（保存）

否则会出现如下问题：

NEWGAL>FBCON: Can't open /dev/tty0: No such file or directory

NEWGAL: Set video mode failure.

InitGUI: Can not initialize graphics engine!

8. 制作根文件系统

#cd /root/minigui/arm

#mkyaffsimage root_default root_minigui.img

9. 下载并执行 MiniGUI 应用程序

把上面制作好的根文件系统下载并烧写到开发板上，依次执行如下命令：

#cd /tmp

#rz

(把 helloworld 使用 rz 命令下载到开发板的/tmp 目录下)

#chmod 777 helloworld

（修改 helloworld 的可执行权限）

#./helloworld

如果一切顺利，就可以看到 helloworld 的 MiniGUI 图形界面了。

10. 编写 MiniGUI 应用程序

#cd /root/minigui/arm

#mkdir helloworld

#cd helloworld

#cp ../ mg-samples-1.6.10/src/helloworld.c .

（复制源文件）

#vi Makefile

（编写 Makefile，添加内容如下：）

TARGET=helloworld

```
CROSS_PATH=/usr/local/arm/3.3.2/arm-linux
#制定交叉编译函数库的位置
helloworld:$(TARGET).c
        arm-linux-gcc -o $(TARGET) $(TARGET).c \
        -L$(CROSS_PATH)/lib   \
        -I$(CROSS_PATH)/include \
        -lpng -lminigui -lz –lpthread
#-L 选项后添加库文件路径，-I 选项后添加头文件搜索路
#径，后面的-lpng -lminigui -lz –lpthread 分别指名了编译
#时所依赖的函数库，如果使用了 MiniGUI 的扩展功能，
#还要添加-lmgext。
clean     :
        rm $(TARGET) *.o -rf
```

（保存）

#make

把生成的 helloworld 下载到开发板上执行，会发现和前面的结果一样，这里只是为了说明如何编译自己编写的应用程序，具体的编程方法参考 MiniGUI 的用户手册，上面写的已经十分详尽。

注：此文档由网上搜集资料和个人实际操作后总结整理而出，十分感谢那些无私奉献的网友。但个人觉得不够完善和全面，特总结至此，希望对大家有所帮助。

其中可能还有不少错误，欢迎批评指正！