基于 ARM 的嵌入式 Linux 图形界面的研究与实现①

许 勇[1]，陈蜀宇[2]

[1](重庆大学 计算机学院，重庆 400030)
[2](重庆大学 软件学院，重庆 400030)

摘 要：主要针对基于 ARM 平台的嵌入式 Linux 图形界面的研究与实现。通过裁剪定制在 ARM9 上构建一个基本的 Linux 系统，然后在此 Linux 系统上移植了 X 协议和一个小型的窗口管理器 Matchbox，最后在此之上移植了 GTK 图形库和基于 GTK 的应用程序。

关键词：嵌入式；Linux；ARM；GTK

Research and Implementation of the Embedded Linux Graphic on ARM

XU Yong[1], CHEN Shu-Yu[2]

[1](Computer Science Department, Chongqing University, Chongqing 400043, China)
[2](Software Engineer Department, Chongqing University, Chongqing 400043, China)

Abstract: This paper mainly researches the embedded linux graphic. In this paper, we first construct a basic Linux system on the ARM9, then transplant X protocol into the system. After that, a small window manager—MatchBox is transplanted into this system, too. At last, we transplant the GTK+ graphics library and the GTK+ applications.

Key words: embedded system; Linux; ARM; GTK

1 引言

传统的嵌入式系统由于硬件资源方面的限制，往往没有图形界面或者只有很简单的图形界面。近年来以 ARM，MIPS 等为主的嵌入式处理器性能的提高，为嵌入式软件的发展提供了必要的基础。Linux 作为一个非常优秀的开源操作系统已经被广泛应用到大多数常见的处理器上。桌面 Linux 的图形界面采用基于 X 协议的图形库，一般可分为两种：第一，基于 GTK 图形库的 GNOME；第二，基于 KDE 库的 QT。以往嵌入式 Linux 的图形界面往往是采用没有 X 协议的嵌入式 QT 图形界面。但 QT 是商业模式，并且其嵌入式应用程序和桌面的应用程序开发有较大的差异，程序的移植性不好。

本系统硬件平台采用 ARM9 内核的 S3C2440 处理器，该处理器硬件性能可以运行基于 X 的图形界面。由于基于 X 的桌面系统软件非常丰富，其经过少量修改便可移植到基于 X 的嵌入式 Linux 系统中。从而可以大大缩短了开发周期和成本，并且可以运用于各种嵌入式终端。

本论文首先在开发板上构建了一个基本的 Linux 系统。然后通过 Scratchbox 构建了一个模拟本地开发的交叉编译环境用于后面的的 X 协议的移植和 GTK+ 库的移植，最后通过修改少量源码，移植了桌面系统的应用程序至本系统中。

2 嵌入式Linux系统构建

嵌入 Linux 系统软件的构成一般分成四个部分 Bootloader、Linux 内核、根文件系统、图形界面。其中前三个是系统运行所必须的部分，图形界面则一般根据实际应用的情况来决定。本节将从 Bootloader 移植、Linux 内核裁剪定制、根文件系统构建三个方面进行阐述如何构建一个基本的 Linux 系统。

2.1 Bootloader 移植

Bootloader 所起的作用是当系统上电以后初始化

① 基金项目:重庆市自然科学基金(CSTC2010AB2001)
收稿时间:2011-02-18;收到修改稿时间:2011-03-19

各个硬件的各个模块，例如首先会初始化时钟，关闭中断，配置 SDRAM，以及获取其他硬件信息，并且以某种格式将操作系统所需要的各种信息存放在一个固定的地址供操作系统读取。

Bootloader 种类非常多，常见的有 U-Boot，GRUB，Vivi 等，其中 U-Boot 在嵌入式范围内使用较多，Vivi 是三星正对其自己的 ARM 芯片设计的引导程序，GRUB 在 PC 机上用得非常广泛，Ubuntu、Fedora 等都是用 GRUB。

本系统中采用 U-Boot 1.6 版本作为 Bootloader，它直接支持 S3C2410 芯片，要移植到 S3C2440 只需做一些小改动，例如时钟设置，NANDFlash 读写设置等。

在 U-Boot 顶层 Makefile 添加目标板的信息(目标板的名称定义为 xy24x0)：

xy24x0_config: unconfig

@$(MKCONFIG) $(@:_config=) arm arm920t xy24x0 NULL s3c24x0

然后在 Board 文件夹下面新建相应的目标板信息 xy24x0，其内容参照 SMDK2410,修改 SDRAM 配置信息，系统时钟设置，NANDFlash，网络等做相应的修改[1,2]。修改完成后通过交叉编译生成 U-boot.bin 文件。

make xys3c24x0_config

make

编译成功后会在 U-Boot 的根目录下产生 U-boot.bin，使用 sjf240x 软件通过 JTAG 接口烧写至目标板 NOR Flash.的零地址。

2.2 Linux 内核移植

Linux 是可以裁剪、功能非常强大的开源操作系统，包含进程调度，进程通信，文件系统，网络等模块。

本系统采用 Linux2.6.22 版本，其主要修改容如下：在顶层 Makefile 添加以下内容

ARCH?=$(SUBARCH)

CROSS_COMPILE?=

ARCH?=arm

CROSS_COMPILE?=arm-linux-

然后再源码根目录执行 make menuconfig 通过图形界面对 Linux 进行配置和裁剪[2-4]:如图 1 所示。

最后在根目录执行 make uImage 既可以生成内核文件，在此编译过程中可能会出现一个错误，原因是 uImage 是按 U-Boot 启动格式的内核文件，需要将 U-Boot 下的 mkimage 拷贝至 Linux 内核根目录下的 scripts 文件夹下才能成功生成 uImage。通过 tftp 将 uImage 下载至开发板 RAM 中，并烧写至 Nand Flash 中的 0 地址。其中 uImage 大小一般不超过 2M，所以只用擦出前 2M 的地址。具体操作如下：

tftp 0x32000000 uImage

nand erase 0 0x20000

nand write.jffs2 0x32000000 0 $(filesize)

设置 U-boot 环境变量

set bootcmd ‘nboot 0x32000000 0 0;bootm 0x32000000’

saveenv

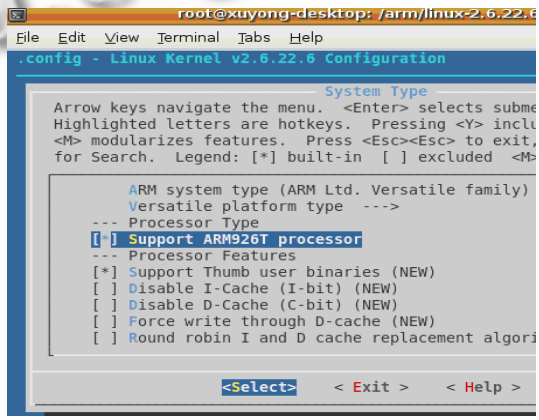

图 1　Linux 内核裁剪与定制

2.3 根文件系统的构建

只有 Linux 内核还不能成为一个完整的系统，必须还有根文件系统、配置文件、基本命令等。.通常一个基本的根文件系统应该包括 shell 命令，配置文件，设备文件，必要的库文件系统。根文件的构建，就是创建这些文件和目录。例如在/bin、/sbin 下的各种可执行文件 /etc 下的各种配置文件等。BusyBox 是一个开源项目，并且它将 Linux 很多命令集合到一个很小的可执行文件中。通过 BusyBox 创建根文件系统，只需要在/dev 目录下创建必要的设备节点，/etc 目录下创建一些配置文件即可[5]。

编译安装 BusyBox：

#tar xjvf busybox-1.7.0.tar,bz2

cd /busybox-1.7.0

make menuconfig

然后修改 Busybox 根目录下的 Makefile 文件，ARCH ？=arm

cross_COMPILE ?=arm-linux-

#make CONFIG_PREFIX=/nfs_root install

即可在 nfs_root 目录下生成根文件系统的基本文件。

为了方便开发在整个移植过程中只将 U-boot 和 Linux 内核烧写入 Flash。文件系统包括后面移植的图形库等都通过 nfs 网络文件系统映射到开发板。并设置如下：

setenv bootargs console=ttySAC0 root=/dev/nfs nfsroot=192.168.0.10:/nfs_root ip=192.168.0.9:192.168.0.10:192.168.0.1:255.255.255.0 mem=30M

至此一个基本的 Linux 系统已经构造完成，上电即可运行。

3 嵌入式linux图形界面的移植

3.1 Scratchbox 开发环境的构建

前面构建基本 Linux 系统直接使用交叉编译工具 arm-linux-gcc 即可，在本节及以后的图形界面移植有很多依赖软件，所以采用模拟目标系统的开发环境方式来开发，以方便依赖软件的安装和图形库的移植。

Scratchbox 是一个提供模拟目标系统的开发开发环境的跨平台开发工具，它提供了完整的集成工具链以用来跨平台编译集成出一个 Linux 发行版[6]。

Scratchbox 主要有两个功能：

① 包装交叉工具链，以“本地工具链”的形式运行。

② 在主机上模拟目标机器运行。

Scratchbox 主要有以下核心文件 scratchbox-core, scratchbox-devkit-cputransp,scratchbox-devkit-debian,scratchbox-devkit-doctols,scratchbox-devkit-perl,scratchbox-libs,scratchbox-toolchain-host-gcc. Scratchbox 安装：

首先使用以下命令将其文件解压到/scratchbox 中：

for f in $(ls *.tar.gz); do tar xzf $f –C /; done;

运行 Scratchbox，第一次运行需要进行一些设置，默认即可。然后增加一个用户 xy 并登陆，命令如下：

#sudo /scratchbox/run_me_first.sh

#sudo /scratchbox/sbin/sbox_adduser xy

#sudo /scratchbox/sbin/sbox_ctl start

$sudo /scratchbox/login

然后安装交叉编译工具链，解压 scratchbox-arm-linux-gcc.tar.gz

至/scratchbox/compilers 中即可

安装其它依赖软件：

① 安装 ncurses

② 安装 libtool

登陆 scratchbox 以后设置环境变量 >export GCC_EXEC_PREFIX=/usr/lib//

这样才能让编译器正确找到路径。到此为止 scratchbox 开发环境全部完成[7]。

3.2 X 的移植

Linux 的图形界面完全是在用户态实现：包括窗口的管理，桌面，文件浏览器等，都是由一个应用程序来实现的。X 协议就是来管理这些应用程序的。当很多程序需要操作屏幕，键盘，鼠标等输入输出设备时则由一个名为 X Server 的程序来统一管理，其他的应用程序被称为 X Client。X 是一种协议，实现 X 协议的代码有很多种，其中 XFree86 和 Xorg[8]是使用最广泛的。

下面分别介绍各个依赖软件的作用，并以 zlib 安装为例：

zlib：通用压缩解压函数库

tar xzv zlib-1.2.3-tar.gz

cd zlib-1.2.3

./configure –shared –prefix=/usr

Make

make install

其余安装方法与 zlib 安装方法类似。

libpng：png 图形编码解码程序库

expat：解析 xml 的 C 语言库

freetype：跨平台的字体绘制引擎

libxml-2.0：fontconfig 依赖的软件

fontconfig：用于字体的配置和访问

libdrm：提供直接访问显示设备的接口

openssl：编译 X server 时用到的库

本系统移植 Xorg 做为 X 协议的实现，在网站上可以下到 Xorg 最新的源码,然后根据实际开发板的硬件，修改其参数和去掉一些不用的软件包。主要修改如下：

① 修改 util/modular 文件夹中的 build.sh 文件其中 build_lib()表示对哪些库进行编译。

```
build_lib()
{
build lib libxtrans
build lib libxau
build lib libXdmcp
}
```

② 修改支持开发板 LCD 的分辨率,修改 xserver/hw/kdrive/src/kmode.c。例如分辨率是 320 X 240 则按照其它的格式添加如下信息。其中用到的参数只有前三个。

```
{
320, 240, 64,  16256,
}
```

修改完以上文件以后则进行编译

```
#cd  Xorg
#/util/modular/build.sh –b /usr
```

执行以上命令以后，编译的结果放在 scratchbox 里面的 usr 文件夹下。里面包含了 X 的所有部件：库，应用程序，X server，字库，头文件等[7]。

将以上内容拷贝至 nfs_root 目录，则 X 协议移植完毕。

3.3 Matchbox 的移植

Matchbox 为嵌入式设备提供了一个开放源代码的基于 X 协议的 GUI 环境，相比 GNOME，KDE 等环境具有小巧，易于裁剪等功能。Matchbox 核心是一个小型的窗口管理器，是基于 PDA 风格的[7]。

① 安装依赖的库 jpeg 库

编译前先修改其 configure 文件，添加如下类容

```
$srcdir/ltconfig
$disable_shared
$disable_static
$srcdir/ltmain.sh arm
```

然后配置并编译

```
>./configure–enable-shared–enable-static–prefix=/usr
>make
>make install
```

② 编译并运行 matchbox:

修改文件 matchbox-autobuild.sh:

DEST 修改成”/usr”，该值作为安装时候的路径。

在 scratchbox 中运行 matchbox-autobuild.sh 即可编译 scratchbox。然后将编译的结果复制到前面制作的根目录下，并改变其读属性：

```
cp –rf /scratchbox/users/xy/targets/arm/usr /nfs
$sudo chown book:book nfs –R
构建字符目录
cd /nfs_root
ln –s /usr/lib/X11/fonts usr/share/
```

运行 Matchbox，使用脚本 matchbox-session 启动所有程序[7,9]：

```
# Xfbdev –mouse mouse –keybd keyboard &
# export DISPLAY=:0
# export HOME=/root
# matchbox-session &
……
```

LCD 屏幕上会显示图像，则 matchbox 移植成功。

3.4 GTK 库的移植

GTK 一般用来泛指 GTK。如果将 GTK+和 GTK 比较，则 GTK 表示以前的 GTK 库，GTK+表示目前应用比较广泛的新 GTK 库。本系统则采用 GTK+库。(在本论文中，如未特别说明的 GTK 库是指广泛意义上的 GTK 库，并非是和 GTK+相对的旧版本的 GTK 库)

GTK+的结构如图 2 所示：

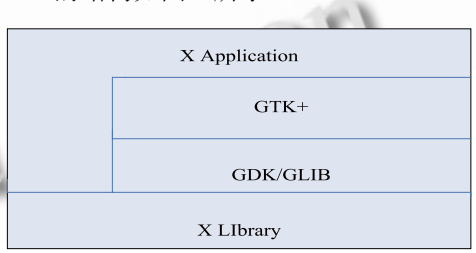

图 2　GTK+图形库的层次关系

X library 则直接与 X server 通信，GLIB 是 GTK+的基础底层核心库。GDK 是 X library 函数调用的一个基本封装。GTK+不会直接与 X 通信，而是通过中间层 GDK。GDK 屏蔽了不同窗口的系统差异，提高了移植性[7]。

① GTK+的依赖库及其安装

Glib C 语言函数库

Cairo 二维图形库

Pango 国际化文本处理

ATK 可访问工具箱

以上软件类似于 3.2 节的 zlib，以上依赖库安装好以后就可以安装 gtk+库了。

② 编译安装 GTK+库

>tar xjf gtk+-2.10.9.tar.bz2

>cd gtk+-2.10.9/

>./configure –prefix=/usr

>make

>make install

最后编译的结果存放在 scratchbox 的/usr 里面，将以上/usr 中的文件复制到 nfs_root 的/usr 中，至此 GTK+库已经移植完毕。此时可启动系统，系统启动后的界面如图 3 所示：

图 3 基于 GTK 库的图形

4 实验结果

本节移植基于 X 和 GTK 库的应用程序 xterm，这个应用程序直接可以运行于基于 GTK 库的桌面 Linux 系统之上。

Xterm 是一个 X Window 上的标准虚拟终端。用户可以在同一个显示器上开启许多 xterm，每一个都为其中运行的进程提供独立的输入输出。通过简单的配置和编译即可使 xterm 运行于本系统上。

>tar xzf xterm.tar.gz

>cd /xterm-229/

>./configure–x-includes=/usr/include–x-libraries=/usr/lib –prefix=/usr

>make

>make install

将其解压后复制到/nfs_root/usr/bin 目录下,即可启动。

本实验成功地移植了 X 协议和 GTK+库至 ARM9 处理器上，最后将 xterm 成功地运行在该系统之上。如图 4 所示：

图 4 xterm 运行结果

5 结语

本论文将 Linux 系统的移植和构建作为一个基础的部分，重点研究了在嵌入式 Linux 系统上移植 X 协议和 GTK 库。最后通过少量修改移植了一个在桌面 Linux 系统应用非常广泛的应用程序 xterm。而基于 QT 的应用程序移植要比这个复杂的多，其开发成本和时间都要多于基于 GTK 库的应用程序。目前从嵌入式 Linux 所发展的程度来看，X 和 GTK 运行在嵌入式 Linux 之上是发展的必然趋势。

参考文献

1 冷玉林,钟将.基于 ARM 的嵌入式 Linux 系统构建.计算机系统应用,2010,19(11):23−26.

2 李宗海,陈蜀宇,李海伟.嵌入式 Linux 系统在 ARM 平台上的构建.计算机系统应用,2010,19(10):153−157.

3 商斌.Linux 设备驱动开发入门于编程实践.北京:电子工业出版社,2009.

4 陈莉君.Linux 内核设计与实现.第 2 版.北京:机械工业出版社,2006.

5 韩存兵.构建嵌入式 Linux 系统.北京:中国电力出版社,2004.

6 Scratchbox. http://www.scratchbox.org/

7 韦东山.嵌入式 Linux 应用开发完全手册.北京:人民邮电出版社,2008.

8 Xorg.[2010-12-21]. http://www.x.org/wiki/Matchbox, [2007], http://matchbox-project.org/