嵌入式最小 Linux 的移植及系统性能测试*

董 华[1] 苗 晟[2,3]
(1.云南省人民检察院信息网络处 昆明 650228;2.西南林业大学计算机与信息学院 昆明 650224;
3.中国科学院天体结构与演化重点实验室 昆明 650011)

摘 要:最小 Linux 也称最简系统,是基于 Linux 内核构建的能够在嵌入式系统上运行的无界面最小系统。通过最小系统的搭建和移植有利于快速掌握嵌入式系统的系统移植和编程的基本方法。针对目前较新的基于 Cortex-a9 架构设计的 Exynos4412 平台,介绍最小 Linux 系统的交叉编译环境搭建及相关驱动软件的配置,内核源码以及 Busybox 的编译和镜像文件的烧写移植流程,并给出主要移植步骤和相关指令介绍。在最小系统基础上,编写测试代码来测试嵌入式最小 Linux 系统下软件的运行时间以分析 Cortex-a9 架构的性能,通过实验表明,基于四核 Cortex-a9 架构的最小 Linux 系统百万次除法运算时间约为 0.18 s,能够满足大多数嵌入式系统性能指标。

关键词:嵌入式系统;最小 Linux 系统;Cortex-a9;内核源码
中图分类号:TN40 **文献标识码**:A **国家标准学科分类代码**:510.10

Transplantation and Testing of Embedded Minimum Linux System

Dong Hua[1] Miao Sheng[2,3]
(1. Information Network Division of the People's Procuratorate of Yunnan Province, Chinese Academy of Sciences, Kunming 650011, China; 2. Computer and information College of Southwest Forestry University, Kunming 650224, China; 3. Key Laboratory for the Structure and Evolution of Celestial Objects, Kunming 650228, China)

Abstract: The minimum Linux system, which is running on the embedded system whiout interface building by the Linux kernel. Through the minimum system establishment and transplantation, it is helpful to grasp the basic method of system transplantation and programming. According to the Cortex-a9 framwork designing of the Exynos4412 platform, Building cross-compiler environment and some related software is introduced, paper focus on the the kernel source code and Busybox compiler and the flow of the transplantation, the main steps of transplant and main instruction were also introduced. The test code has been programmed to show the performance of the Cortex-a9 and minimum Linux system. The experimental results show that million times the division operation time spend about 0.19s, which can meet the performance requirements of most embedded systems.

Keywords: embedded system; minimum Linux system; cortex-a9; kernel source

0 引 言

随着应用服务需要的日益复杂和物联网技术的发展,很多高端嵌入式系统都需要使用操作系统开发[1]。除了一些对电路和体积要求较高的应用[2-3],嵌入式操作系统对嵌入式系统开发带来了很大便捷性和灵活性[4-5]。构成一个好的嵌入式系统需要硬件平台、操作系统、应用软件相互协调和配合,缺一不可[6-7],随着的硬件平台和架构的诞生,相应的操作系统也在不断更新。

Linux 是嵌入式系统中市场占比最大的操作系统(2015 年超过 50%),现在有很多文章或技术文档介绍 Linux 系统在 ARM9 架构上的移植[5-8],但 ARM 公司早已在 2012 年就推出了 Cortex 系列,而目前很多基于 ARM 的高端应用平台也大量更新为 Cortex-a 系列,其有着更高效的运算能力。而目前介绍在 Cortex-a 系列上 Linux 的移植和开发的相关文章却不多。

本文主要叙述基于采用四核 Cortex-a9 架构的三星 Exynos4412 为平台,探讨相关交叉编译环境的搭建,最小 Linux 系统编译和移植基本方法,并通过 Linux 系统编程实验对比测试基于 a9 的微处理器的性能,验证最小 Linux 系统的稳定性和实用性。

收稿日期:2017-02
* 基金项目:中科院云南天文台开放课题基金(OP201506)、西南林业大学教改基金(yb201501)项目资助

1　Linux 最小系统搭建

1.1　Linux 交叉编译环境搭建

交叉编译环境的搭建是嵌入式系统开发的基础，但是搭建 Linux 交叉编译环境和单片机开发环境并不一样。没有一个类似 Keil 的集成开发环境，常用方法是在上位机上安装虚拟机并运行 Linux 系统，在 Linux 环境下使用交叉编译器对源码进行编译，再将编译生成的镜像文件通过 fastboot 工具烧写到目标板中，就可以上电运行了。首先介绍交叉编译环境的搭建，包括：

1）虚拟机及 Linux 系统发行版

在上位机上搭建交叉编译环境首先需要安装 Linux 系统，但现在普遍做法是安装虚拟机，例如 VMware，在此基础上再选择一款 Linux 发行版进行安装。

2）ADB 驱动和 fastboot 工具

Adb 驱动是调试硬件平台的有力工具，尤其是针对 Android 系统。镜像文件的烧写和移植离不开 Adb 驱动的安装。驱动正常安装后就可以使用 fastboot 工具连接设备了，这对于后续烧写镜像文件是十分重要的。

3）串口超级终端助手

最小 Linux 系统是无界面的，所以其调试命令输入和结果输出一般都采用串口通信，因此串口超级终端也是调试过程中的有力工具。它既可以发送命令，例如分区格式化就可以通过超级终端完成，也可以反馈目标板的信息。

1.2　Linux 最小系统搭建

Linux 系统架构如图 1 所示。其中 fastboot 是嵌入式操作系统移植不可或缺的，但是该部分很多厂家在目标板生产时就已经移植好了，多数不需要用户自己烧写，所以系统移植主要是针对内核和文件系统。

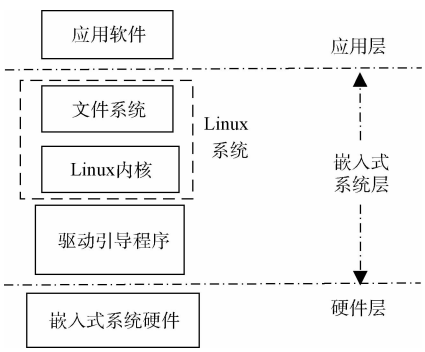

图 1　Linux 嵌入式系统架构

1）Linux 内核

内核是操作系统的核心，具有很多操作系统最基本功能，它负责管理系统的进程、内存、设备驱动程序、文件和网络系统，决定着系统的性能和稳定性，其架构如图 2 所示。

这里强调一下，文件系统作为整个系统的一部分，也受内核管理。Linux 内核由：内存管理、进程管理、设备驱动程序、文件系统和网络管理等部分组成。

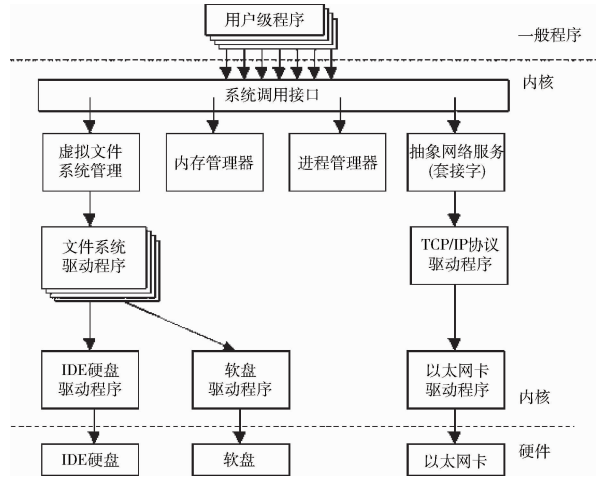

图 2　Linux 内核架构

2）内核裁剪和 .config 文件生成

嵌入式系统裁剪是嵌入式系统移植的重要内容之一，一般普通的 Linux 内核编译后大概有 70～80 M，这么大的内核对于 PC 来说不是问题，但是对于资源有限的嵌入式系统来说，那可就很大了。因为内核很大一部分都是常驻内存的，所以对于内存和硬盘来说，都是一种浪费。内核裁剪一般使用 # make menuconfig 命令，目的都是生成目的都是生成一个 .config 文件，所以 menuconfig 只是一个调试工具。

跟 make menuconfig 这个命令相关的文件，包括 3 类，有 .config，Kconfig 和 Makefile。这是几类文件各有作用。Kconfig 和 Makefile 是配合使用的，目的是生成 .config 文件，.config 指明了在内核编译的时候那些内容要编译进内核，哪些是不需要的。而 makefile 文件是脚本文件，既使用 make 编译命令的时候就依次执行 makefile 文件语句，完成编译。这些配置文件内容比较繁杂，相互交融，可参考文献[10]。

源码是需要编译器进行编译的，在交叉编译中使用 arm-none-linux-gnueabi-gcc 工具进行编译，编译路径的设置可以指明编译器的位置，例如，用 vim 命令打开源码中的 makefile 文件，找到里面的 CROSS_COMPILE，其对应的就是内核编译器的路径，在内核编译中执行 make 命令，而 make 命令首先就是执行 makefile 文件，makefile 文件中就通过上述路径寻找需要的编译器。编译路径的设置一定要和解压编译器的路径对应起来，否则系统会找不到编译器而无法完成编译工作。

3）Busybox 设置

搭建嵌入式 Linux 系统移植还离不开一个工具—Busybox，它是一个集成度很高的编译工具。busybox 是一个集成了 100 多个最常用 Linux 命令和工具的软件，还集成了包括 http 服务器和 telnet 服务器，而整个 Busybox 也

就只有 1 M 左右的大小。Busybox 的获取方式一般可以通过网络下载获得,官网的网址是:www. busybox. net。获得的 Busybox 文件是一个在 Linux 系统中压缩的文件,如:Busybox-1. 22. 1. tar. bz2,这种文件和内核源码一样必须在在 Linux 系统中进行解压才能正常使用。

接下来就可以对 Busybox 进行配置,配置 Busybox 和内核配置方式比较类似,都是使用 make menuconfig 命令,输入配置命令后出现配置主页面。– 进入界面"Busybox Settings"→"Build Optiions"→"Cross Compiler prefix"将其配置为"arm-none-linux-gnueabi-"

接下来配置二进制安装文件:

进入界面"Installation Options"→"BusyBox installation prefix"将其配置为

"../system"并保存退出。

下一步是编译 Busybox,使用 make 命令完成编译。然后再使用 make install 命令,将编译好的二进制文件安装到../system 中,这样就完成了 Busybox 的配置。采用 cd..返回上层目录,用 ls 命令可查看生成了的 system 文件,如图 3 所示。

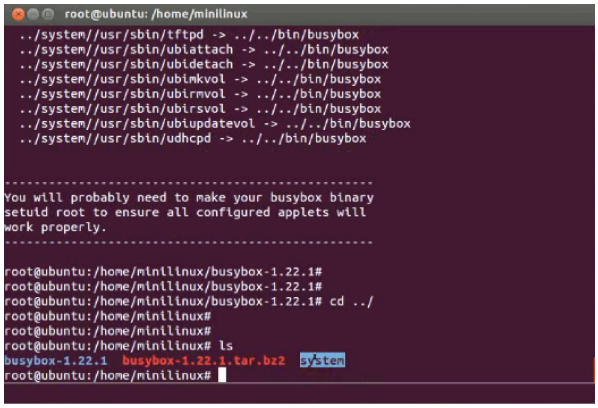

图 3　通过 Busybox 的配置生成 system 文件

最小系统的构建还需要一些其他的文件,分别包括了构成最小系统所必须的各种内容,例如网络驱动等。构建最小系统需要的文件如图 4 所示。

图 4　搭建最小系统需要的其他文件

制作的文件系统还需要放到 dev、etc、lib、mnt、proc、sys、tmp、var 文件夹中,使用命令"mkdir dev etc lib mnt proc sys tmp var"创建文件夹,并将上述文件拷贝到对应文件夹下(使用 ssh 软件拷贝较为快捷)。在所有文件制作完成后使用如下命令就可以生成烧写的镜像文件:

"make_ext4fs -s -l 314572800 -a root -L linux system. img system"

2　最小 Linux 系统移植

以讯为公司的 itop4412 开发平台为例,该平台是基于三星的 Exynos4412 微处理芯片,采用 Cortex-a9 架构设计的。最小 Linux 系统移植的软件涉及到串口超级终端和 ADB 驱动的使用,这是应该在操作平台移植之前完成的前期工作和必要的准备。

最小系统的 Linux 移植就是将以下 4 个二进制镜像文件烧写到开发板的过程,分别是:

"u-boot-iTOP-4412. bin"、嵌入式系统的 bootloader 二进制文件;"zImage",Linux 内核二进制镜像文件;"ramdisk-uboot. img",ramdisk 的启动文件;"system. img",Linux 内核上挂载的文件系统;

烧写过程分为两个步骤,1)在超级终端下使用串口命令完成存储器的分区,2)使用 OTG 接口烧写方式(也叫 fastboot 烧写方式)烧写上述的 4 个文件。在准备好烧写文件及连接好硬件之后,按如下步骤进行操作。

打开超级终端,然后上电启动开发板,按"回车",进入 Uboot 模式,如图 5 所示。

进入 uboot 模式后的串口终端输出

创建 eMMC 分区并格式化。即在串口终端中依次输入执行如下分区命令:

"fdisk -c 0"

fatformat mmc 0:1

ext3format mmc 0:2

ext3format mmc 0:3

ext3format mmc 0:4

fastboot

```
In:    serial
Out:   serial
Err:   serial
eMMC OPEN Success.!!
                        !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.!

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!


Checking Boot Mode ... EMMC4.41
SYSTEM ENTER NORMAL BOOT MODE
Hit any key to stop autoboot:  0
iTOP-4412 #
```

图 5　烧写时串口终端显示界面

注意,fastboot 命令需要与 PC 上的 USB_fastboot_tool 工具配套使用,而且 fastboot 命令需要进入 uboot 模式中才能使用。

至此串口终端命令使用完成,后续需要连接 OTG 线,用 OTG 接口方式烧写内核,这里再强调一下,连接 OTG 线一定需要检查 Adb 驱动是否正确安装,如果没有正确安装的话需要重新完成 Adb 驱动安装。在 Adb 驱动正常情

况下按照如下步骤烧写系统：

在 PC 上运行“USB_fastboot_tool”-->“platform-tools”文件夹中的文件

“cmd.exe”如图 6 所示。

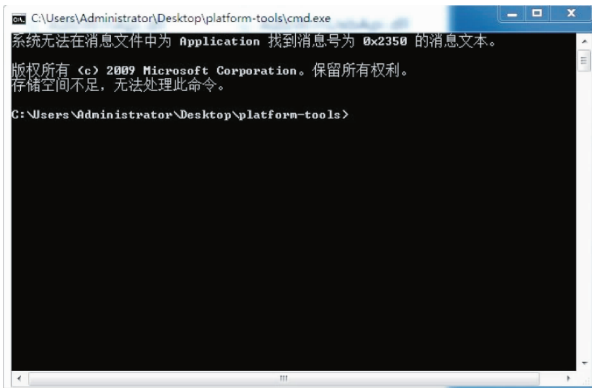

图 6　fastboot 工具的 cmd 对话窗口

在上述窗口下依次输入以下命令完成系统烧写：

fastboot.exe flash bootloader u-boot-iTOP-4412.bin// Uboot 烧写

注意：厂家一般不建议用户烧写引导驱动文件“u-boot-iTOP-4412.bin”，可跳过此步骤，因为出厂前已经烧写过这个镜像文件了。

fastboot.exe flash kernel zImage// zImage 内核烧写

fastboot.exe flash ramdisk ramdisk-uboot.img// ramdisk 内核烧写

fastboot.exe flash system system.img// system 文件系统烧写

fastboot-w//擦除命令

fastboot reboot//重启开发板命令

至此，最小 Linux 系统烧写完毕，烧写完毕后重新断电启动开发板，即可进入 Linux 最小系统进行系统编程开发。

3　系统性能测试

基于 ARM 的 Cortex-a9 架构处理器上移植的最小 Linux 构成的嵌入式系统有着结构小巧，运算速度快等优势，为验证最小 Linux 运行的效果，通过系统编程进行测试。

编写一个简单的 Helloworld 程序进行最小系统测试，源码如下：

```
#include <stdio.h>
main()
{printf("Hello World! \n");}
```

在超级终端上输入执行命令，通过超级终端执行后显示效果如图 7 所示。

为了进一步测试处理器运算速度，采用时间函数对比 a9 系列上最小 Linux 的运算速度。编写时间测试函数，主

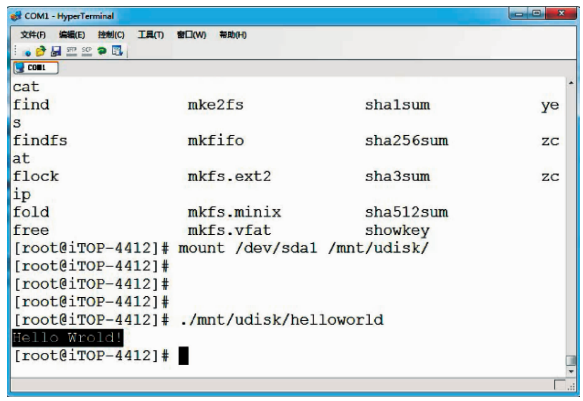

图 7　最小 Linux 系统下执行 Helloworld 程序

函数源码如下：

```
#include<time.h>
#include<sys/time.h>
#include<stdio.h>
main()
{
struct timeval tpstart,tpend;
float timeuse;
gettimeofday(&tpstart,NULL); //记录开始时间
function();
gettimeofday(&tpend,NULL); //记录结束时间
timeuse=1000000*(tpend.tv_sec-tpstart.tv_sec)+tpend.tv_usec-tpstart.tv_usec; //计算差值
timeuse /= 1000000;
printf("Used Time:%f\n",timeuse);
}
```

在上述 function 程序段中可以添加任何希望测试的代码，而通过主函数，可以计算出该段代码运行的时间，例如加一段百万次除法的运算代码，通过编译执行，串口超级终端显示如图 8 所示。

图 8　执行例程串口终端的显示结果

从图 8 中可以看出，在四核 a9 平台的最小 Linux 平台下，百万次除法的耗时约为 0.19 s。此外再通过1 024 点 fft 的运行速度来看，1 024 点 fft 在最小 Linux 系统下运行耗时约为 1.13 ms 左右，虽然这个计算是在系统运行了 Linux 操作系统的计算速度，真实的 Cortex-a9 绝对运算速度应该更快，但是对比 Stm32 的 1 024 点 fft 运算速度（官方公布的为 72 M 芯片运算时间为1.768 ms），基于四核 Cortex-a9 的 Exynos4412 微处理还是更有优势。

（下转第 211 页）