嵌入式Linux移植

培训目的

了解嵌入式系统移植方法，理解各系统组成部分的移植方法

培训对象

软件研发

培训讲师

方建江

培训课时

2小时

学习重点

1. 交叉编译
2. Boot/Kernel移植
3. 文件系统构建
4. 应用程序移植

嵌入式Linux系统的移植主要内容：

交叉编译工具链、Boot、Linux内核、文件系统

交叉编译通俗地讲就是在一种平台上编译出能运行在体系结构不同的另一种平台上的程序。

Boot是在系统上电时开始执行，初始化硬件设备，准备好软件环境，然后才调用Linux操作系统内核。

文件系统是Linux操作系统中用来管理用户文件的内核软件层。文件系统包括根文件系统和建立于Flash内存设备之上文件系统。根文件系统包括系统使用的软件和库，以及所有用来为用户提供支持架构和用户使用的应用软件，并作为存储数据读写结果的区域。

交叉编译工具链(cross compile toolchains)

- 交叉编译工具链是一个由编译器、链接器和解释器组成的综合开发环境，一般是用kernel+gcc+glibc+binutils的源码包来编译安装
- 构建交叉工具链的方法

1. 分步编译和安装交叉编译工具链所需要的库和源代码，最终生成交叉编译工具链【比较困难，适合想深入学习构建交叉工具链的开发人员。如果只是想使用交叉工具链，建议使用现成构建好的交叉工具链】

2.通过Crosstool脚本工具来实现一次编译生成交叉编译工具链【该方法相对于方法一要简单许多，并且出错的机会也非常少，大多数情况下使用该方法构建交叉编译工具链】

3.直接通过网上下载已经制作好的交叉编译工具链【简单省事，但局限性太大，构建所用的库以及编译器的版本也许并不适合你要编译的程序，同时也许会在使用时出现许多莫名的错误，慎用】

使用buildroot构建工具链

Buildroot是一个包含Makefile和patch的工具集，可以方便的为你的目标构建交叉工具链、根文件系统以及Linux内核镜像。

1. 下载buildroot源码包,安装启动配置工具所需的包
2. 在/opt/创建工作目录（一般工具链都会放到/opt目录，作为公用），进入工作目录进行配置

   make menuconfig

   配置项有很多，比如用于选择目标的架构、大小端模式、内核版本、gcc版本、ulibc版本等
3. Make编译完成后工具链最终会在工作目录下生成

```
[fangjianjiang@TWSH-EoC buildroot-2009.08]$ ll /opt/toolchains_qca953x
total 20
drwxr-xr-x  5 root root 4096 Oct  8 16:43 .
drwxr-xr-x 13 root root 4096 Oct  8 17:18 ..
drwxr-xr-x  2 root root 4096 Oct  8 16:43 bin
-rw-r--r--  1 root root    0 Oct  8 16:43 .fakeroot.00000
drwxr-xr-x  2 root root 4096 Oct  8 16:58 lib
drwxr-xr-x 10 root root 4096 Oct  8 17:09 usr
```

Uboot

Uboot是德国DENX小组的开发用于多种嵌入式CPU的bootloader程序， UBoot不仅仅支持嵌入式Linux系统的引导，当前，它还支持NetBSD， VxWorks， QNX， RTEMS， ARTOS，LynxOS嵌入式操作系统。Uboot除了支持PowerPC系列的处理器外，还能支持MIPS、 x86、ARM、NIOS、XScale等诸多常用系列的处理器。

主要功能就是为了启动内核，它将内核从flash中拷贝到ddr中，然后跳转到内核入口中，交由内核控制权

Uboot的目录结构

```
[uboot@localhost u-boot-1.1.4]#tree -L 1 -d
.
|-- board
|-- common
|-- cpu
|-- disk
|-- doc
|-- drivers
|-- dtt
|-- examples
|-- fs
|-- include
|-- lib_arm
|-- lib_generic
|-- lib_i386
|-- lib_m68k
|-- lib_microblaze
|-- lib_mips
|-- lib_nios
|-- lib_nios2
|-- lib_ppc
|-- net
|-- post
|-- rtc
`-- tools
```

2. board：和一些已有开发板有关的文件. 每一个开发板都以一个子目录出现在当前目录中，比如说:SMDK2410, 子目录中存放与开发板相关的配置文件.

3. common：实现 u-boot 命令行下支持的命令，每一条命令都对应一个文件。例如 bootm 命令对应就是 cmd_bootm.c。

4. cpu：与特定 CPU 架构相关目录，每一款 U-boot 下支持的 CPU 在该目录下对应一个子目录，比如有子目录 arm920t 等。

5. disk：对磁盘的支持。

5. doc：文档目录。U-boot 有非常完善的文档，推荐大家参考阅读。

6. drivers：U-boot 支持的设备驱动程序都放在该目录，比如各种网卡、支持 CFI 的 Flash、串口和 USB 等。

7. fs: 支持的文件系统，U-boot 现在支持 cramfs、fat、fdos、jffs2 和 registerfs。

8. include：U-boot 使用的头文件，还有对各种硬件平台支持的汇编文件，系统的配置文件和对文件系统支持的文件。该目录下 configs 目录有与开发板相关的配置头文件，如 smdk2410.h。该目录下的 asm 目录有与 CPU 体系结构相关的头文件，asm 对应的是 asm-arm.

9. lib_xxxx: 与体系结构相关的库文件。如与 ARM 相关的库放在 lib_arm 中。

10. net：与网络协议栈相关的代码，BOOTP 协议、TFTP 协议、RARP 协议和 NFS 文件系统的实现。

11. tools：生成 U-boot 的工具，如：mkimage, crc 等等。

Uboot移植

1. 运行的两个阶段简要说明：

U-Boot代码一般分为stage1和stage2两大部分。stage1依赖于cpu体系结构如设备初始化代码，常用汇编语言编写以达到短小精悍,提高系统运行效率的目的。它主要包括cpu/XXX目录下的start.s。stage2一般采用C语言编写实现复杂功能，这样代码则具有更好的可读性和可移植性，主要包括lib lib_XXX/board.c文件和common/main.c文件中main_loop函数等。

2. 要点：

2.1 顶层Makefile 文件当中的 ARCH 和 CROSS_COMPILE , Kconfig

2.2 初始环境变量ENV，支持image升级

2.3 获取文件系统及内核镜像地址并校验，准备内核启动参数

Linux Kernel

- 获取Linux kernel

  https://www.kernel.org/pub/linux/kernel/
- 配置和编译Linux内核，对其进行相应的裁剪，修改内核以支持相关的硬件设备
- 修改内核根目录下的的Makefile，指明交叉编译器也可以在配置的时候指定：

  make ARCH=mips CROSS_COMPILE=mips-linux- menuconfig
- 修改Flash分区

  分区表应该跟前面boot传递过来的一致，确保kernel启动后能正确找到文件系统

- Linux Kernel加载分区一般采用MTD的方式，这样也为应用层操作分区数据提供了便利。
- 参考MTK7520方案：drivers/mtd/maps/中的tc3162-flash.c文件：
- 创建MTD分区
- add_mtd_partitions(tc3162_mtd_info, tc3162_parts, tc3162_parts_size);

```
static struct mtd_partition tc3162_parts[] =
{
    {name: "Boot",         offset: BOOT_OFFSET,    size: BOOT_SIZE,    mask_flags:0},
    {name: "Env",          offset: ENV_OFFSET,     size: ENV_SIZE,     mask_flags:0},
    {name: "ImageA",       offset: IMAGEA_OFFSET,  size: IMAGEA_SIZE,  mask_flags:0},
    {name: "ImageB",       offset: IMAGEB_OFFSET,  size: IMAGEB_SIZE,  mask_flags:0},
    {name: "Config",       offset: CONFIG_OFFSET,  size: CONFIG_SIZE,  mask_flags:0},
    {name: "Plugin",       offset: PLUGIN_OFFSET,  size: PLUGIN_SIZE,  mask_flags:0},
    {name: "SECTION_EGIS", offset: EGIS_OFFSET,    size: EGIS_SIZE,    mask_flags:0},
    {name: "rootfs",       offset: IMAGEA_OFFSET + 0x130000, size: IMAGEA_SIZE - 0x130000, mask_flags:0},
};
```

内核加载根文件系统

- Boot传递给Kernel的命令行参数(cmdline)告诉Kernel分区信息及rootfs的位置：

```
# cat /proc/cmdline
console=ttyS0,115200 root=31:7 rootfstype=squashfs ro init=/sbin/init activeImg=B rootfs=0x650000/0x8e0000
    mtdparts=athnor0:128k(Boot),64k(Env),7680k(ImageA),7680k(ImageB),576k(Config),128k(SECTION_EGIS),6
    4k(ART) mem=128M
#
```

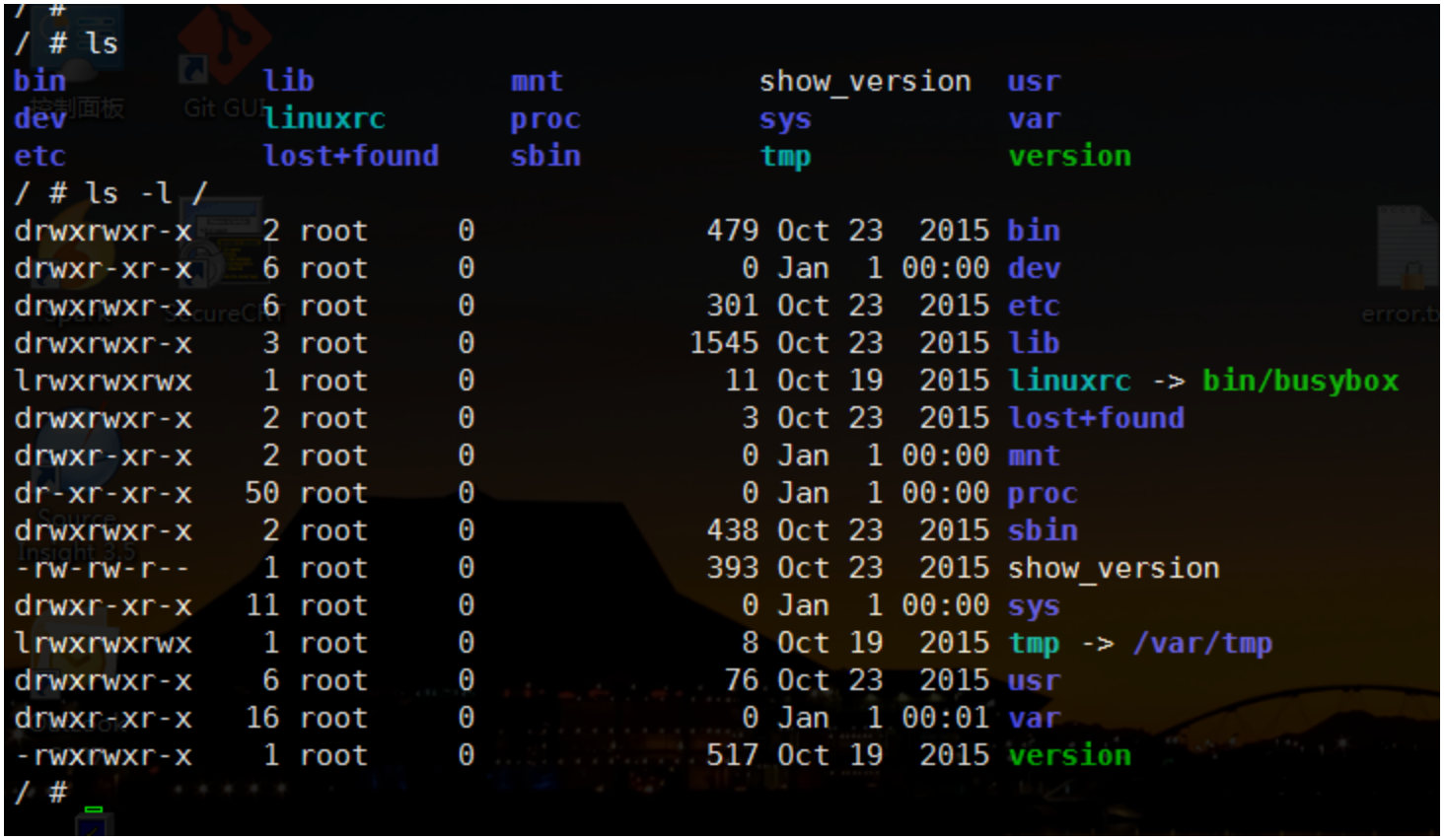
- rootfs参数值格式：

  root=31:7 rootfstype=squashfs ro

  root=/dev/mtdblock5 rootfstype=jffs2 rw

文件系统构建

- S304平台文件系统结构



- 创建Linux运行所需工作目录，比如var、 bin、 sbin、 mnt等
- 构建系统运行所需的库文件

  大部分运行的系统库可以从工具链中直接获取，比如librt.so.0、libuClibc.so，

  用户层的库需要自己去构建，比如HAL，CMC，libssl.so 库
- 启动脚本

  启动脚本一般放在/etc/init.d或者/etc/rc.d目录下，为可执行的shell脚本文件，Kernel启动的第一个进程init进程，会执行rcS脚本，然后由rcS调用 执行其他脚本，加载驱动，初始化网络，挂载配置分区，最终启动应用层进程smd.

Example: rcS

```
#!/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/usr/local/sbin:/usr/S304/bin:/etc/athexport
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/lib/public:/lib/private:/lib/gpl:/lib:/usr/S204/lib:/usr/S304/lib:/var/config/plugin:/var/config/cpm/lib
```

```
# Mounts everything in the fstab

mount -a
```

```
#1. Mount Config partition

CFGMTD=`cat /proc/mtd | grep "Config" | awk -F : '{print $1;}' | sed s/mtd//`

mount -t jffs2 -o sync /dev/mtdblock${CFGMTD} /var/config > /dev/null 2>&1

#2.Init EGIS data

/etc/rc.d/rc.egis

#3.Load drivers

/etc/rc.d/rc.modules

#4.networking init

/etc/rc.d/rc.network

#5.Start S304 system app

/etc/rc.d/rc.S304
```

文件系统类型及创建

- Squashfs与jffs2

  Squashfs的一部分是只读的，占用空间比jffs2小大约20%虽然看起来只会更大，因为重叠了一部分固定的文件，但是在运行后解压出去，但是对于固件本身来说它可以进行压缩，rootfs一般以squashfs类型.

- jffs2看起来一切都是可写的，这样你可以修改甚至启动部分的文件，可以实现高层次的内核修改，一般配置分区是以jffs2格式的.

- 创建jffs2 文件系统实例:

  mkfs.jffs2 --root=$(BUILDFS_PATH) --eraseblock=65536 -b –D dev.txt –squash -o $(OUT_PUT_FS_NAME)

  具体参数选项说明参见命令的帮助

- Squashfs的创建

  文件系统所需要的dev设备必须创建好，否则会导致Kernel无法正常加载，之前遇到一个这样的问题，dev设备创建脚本参考：

  参考命令：

```
mkfs.squashfs $(BUILDFS_PATH) rootfs.squashfs -nopad -noappend -root-owned -comp lzma -b 65536
```

  具体的参数选项可以参考帮助信息，不同的squashfs创建命令使用方法也可能会有所有不同。

应用程序移植

- 使用交叉编译

  CC → CROSS_COMPILE

  LDFLAGS

  工具链里的库和头文件或者

  使用交叉编译出的执行文件可以用Linux Shell命令file检查一下，是否为目标平台的强执行文件，如：
  #file tcpdump
  tcpdump: ELF 32-bit MSB executable, MIPS, MIPS32 rel2 version 1 (SYSV), dynamically linked (uses shared libs), with unknown capability 0x41000000 = 0xf676e75, with unknown capability 0x10000 = 0x70403, stripped

  host=mips

  1/ --

  figure选项要看具体的支持，大多是类似的。

- Install

  Makefile的install目标，注意最后的install路径

Thanks