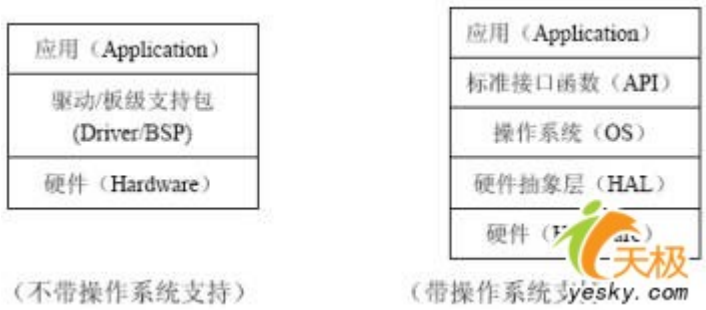
ARM 的嵌入式 Linux 移植体验之操作系统（上）
作者： 宋宝华

摘要

本章介绍了嵌入式 Linux 的背景、移植项目、init 进程修改和文件系统移植，通过这些步骤，我们可以在嵌入式系统上启动一个基本的 Linux。

关键词：ARM，嵌入式 Linux，移植

在笔者撰写的《C 语言嵌入式系统编程修炼之道》一文中，主要陈诉的软件架构是单任务无操作系统平台的，而本文的侧重点则在于讲述操作系统嵌入的软件架构，二者的区别如下图：



嵌入式操作系统并不总是必须的，因为程序完全可以在裸板上运行。尽管如此，但对于复杂的系统，为使其具有任务管理、定时器管理、存储器管理、资源管理、事件管理、系统管理、消息管理、队列管理和中断处理的能力，提供多任务处理，更好的分配系统资源的功能，很有必要针对特定的硬件平台和实际应用移植操作系统。鉴于 Linux 的源代码开放性，它成为嵌入式操作系统领域的很好选择。国内外许多知名大学、公司、研究机构都加入了嵌入式 Linux 的研究行列，推出了一些著名的版本：

·RT-Linux 提供了一个精巧的实时内核，把标准的 Linux 核心作为实时核心的一个进程同用户的实时进程一起调度。RT-Linux 已成功地应用于航天飞机的空间数据采集、科学仪器测控和电影特技图像处理等广泛的应用领域。如 NASA(美国国家宇航局)将装有 RT-Linux 的设备放在飞机上，以测量 Georage 咫风的风速；

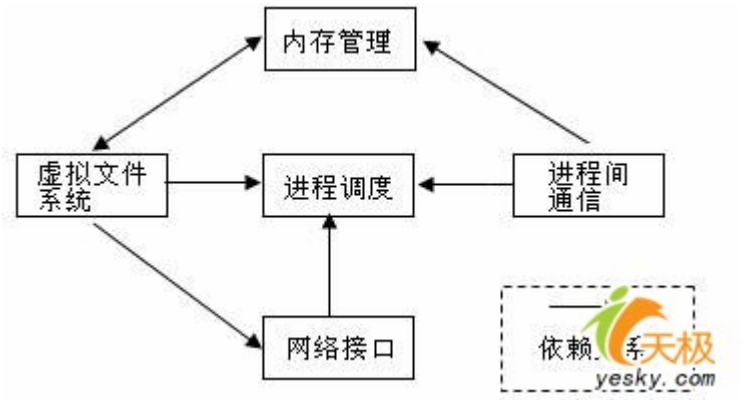
·uCLinux（Micro-Control-Linux，u 表示 Micro，C 表示 Control）去掉了 MMU（内存管理）功能，应用于没有虚拟内存管理的微处理器/微控制器，它已经被成功地移植到了很多平台上。

本章涉及的 mizi-linux 由韩国 mizi 公司根据 Linux 2.4 内核移植而来，支持 S3C2410A 处理器。

1.Linux 内核要点

和其他操作系统一样，Linux 包含进程调度与进程间通信(IPC)、内存管理(MMU)、虚拟文件

系统(VFS)、网络接口等，下图给出了 Linux 的组成及其关系：



Linux 内核源代码包括多个目录：

（1）arch：包括硬件特定的内核代码，如 arm、mips、i386 等；
（2）drivers：包含硬件驱动代码，如 char、cdrom、scsi、mtd 等；
（3）include：通用头文件及针对不同平台特定的头文件，如 asm-i386、asm-arm 等；
（4）init：内核初始化代码；
（5）ipc：进程间通信代码；
（6）kernel：内核核心代码；
（7）mm：内存管理代码；
（8）net：与网络协议栈相关的代码，如 ipv4、ipv6、ethernet 等；
（9）fs：文件系统相关代码，如 nfs、vfat 等；
（10）lib：库文件，与平台无关的 strlen、strcpy 等，如在 string.c 中包含：

```
char * strcpy(char * dest,const char *src)
{
char *tmp = dest;
while ((*dest++ = *src++) != '\0')
/* nothing */;
return tmp;
}
```

（11）Documentation：文档

在 Linux 内核的实现中，有一些数据结构使用非常频繁，对研读内核的人来说至为关键，它们是：

1.task_struct

Linux 内核利用 task_struct 数据结构代表一个进程，用 task_struct 指针形成一个 task 数组。当建立新进程的时候，Linux 为新的进程分配一个 task_struct 结构，然后将指针保存在 task 数组中。调度程序维护 current 指针，它指向当前正在运行的进程。

2.mm_struct

每个进程的虚拟内存由 mm_struct 结构代表。该结构中包含了一组指向 vm-area_struct 结构的指针，vm-area_struct 结构描述了虚拟内存的一个区域。

3.inode

Linux 虚拟文件系统中的文件、目录等均由对应的索引节点(inode)代表。

2.Linux 移植项目

mizi-linux 已经根据 Linux 2.4 内核针对 S3C2410A 这一芯片进行了有针对性的移植工作，包括：

（1）修改根目录下的 Makefile 文件

a.指定目标平台为 ARM：

```
#ARCH := $(shell uname -m | sed -e s/i.86/i386/ -e s/sun4u/sparc64/ -e s/arm.*/arm/ -e s/sa110/arm/)
ARCH := arm
```

b.指定交叉编译器：

CROSS_COMPILE = arm-linux-

（2）修改 arch 目录中的文件

根据本章第一节可知，Linux 的 arch 目录存放硬件相关的内核代码，因此，在 Linux 内核中增加对 S3C2410 的支持，最主要就是要修改 arch 目录中的文件。

a.在 arch/arm/Makefile 文件中加入：

```
ifeq ($(CONFIG_ARCH_S3C2410),y)
TEXTADDR = 0xC0008000
MACHINE = s3c2410
Endif
```

b.在 arch\arm\config.in 文件中加入：

```
if [ "$CONFIG_ARCH_S3C2410" = "y" ]; then
comment 'S3C2410 Implementation'
dep_bool ' SMDK (MERI TECH BOARD)' CONFIG_S3C2410_SMDK $CONFIG_ARCH_S3C2410
dep_bool ' change AIJI' CONFIG_SMDK_AIJI
dep_tristate 'S3C2410 USB function support' CONFIG_S3C2410_USB $CONFIG_ARCH_S3C2100
```

```
dep_tristate ' Support for S3C2410 USB character device emulation'
CONFIG_S3C2410_USB_CHAR $CONFIG_S3C2410_USB
fi # /* CONFIG_ARCH_S3C2410 */
```

arch\arm\config.in 文件还有几处针对 S3C2410 的修改。

c.在 arch/arm/boot/Makefile 文件中加入：

```
ifeq ($(CONFIG_ARCH_S3C2410),y)
ZTEXTADDR = 0x30008000
ZRELADDR = 0x30008000
endif
```

d.在 linux/arch/arm/boot/compressed/Makefile 文件中加入：

```
ifeq ($(CONFIG_ARCH_S3C2410),y)
OBJS += head-s3c2410.o
endif
```

加入的结果是 head-s3c2410.S 文件被编译为 head-s3c2410.o。

e.加入 arch\arm\boot\compressed\ head-s3c2410.S 文件

```
#include <linux/config.h>
#include <linux/linkage.h>
#include <asm/mach-types.h>

.section ".start", #alloc, #execinstr

__S3C2410_start:

@ Preserve r8/r7 i.e. kernel entry values
@ What is it?
@ Nandy

@ Data cache, Intstruction cache, MMU might be active.
@ Be sure to flush kernel binary out of the cache,
@ whatever state it is, before it is turned off.
@ This is done by fetching through currently executed
@ memory to be sure we hit the same cache

bic r2, pc, #0x1f
add r3, r2, #0x4000 @ 16 kb is quite enough...
1: ldr r0, [r2], #32
teq r2, r3
bne 1b
```

```
mcr p15, 0, r0, c7, c10, 4 @ drain WB
mcr p15, 0, r0, c7, c7, 0 @ flush I & D caches

#if 0
@ disabling MMU and caches
mrc p15, 0, r0, c1, c0, 0 @ read control register
bic r0, r0, #0x05 @ disable D cache and MMU
bic r0, r0, #1000 @ disable I cache
mcr p15, 0, r0, c1, c0, 0
#endif

/*
* Pause for a short time so that we give enough time
* for the host to start a terminal up.
*/
mov r0, #0x00200000
1: subs r0, r0, #1
bne 1b
```

该文件中的汇编代码完成 S3C2410 特定硬件相关的初始化。

f.在 arch\arm\def-configs 目录中增加配置文件

g.在 arch\arm\kernel\Makefile 中增加对 S3C2410 的支持

```
no-irq-arch := $(CONFIG_ARCH_INTEGRATOR) $(CONFIG_ARCH_CLPS711X) \
$(CONFIG_FOOTBRIDGE) $(CONFIG_ARCH_EBSA110) \
$(CONFIG_ARCH_SA1100) $(CONFIG_ARCH_CAMELOT) \
$(CONFIG_ARCH_S3C2400) $(CONFIG_ARCH_S3C2410) \
$(CONFIG_ARCH_MX1ADS) $(CONFIG_ARCH_PXA)
obj-$(CONFIG_MIZI) += event.o
obj-$(CONFIG_APM) += apm2.o
```

h.修改 arch/arm/kernel/debug-armv.S 文件，在适当的位置增加如下关于 S3C2410 的代码：

```
#elif defined(CONFIG_ARCH_S3C2410)

.macro addruart,rx
mrc p15, 0, \rx, c1, c0
tst \rx, #1 @ MMU enabled ?
moveq \rx, #0x50000000 @ physical base address
movne \rx, #0xf0000000 @ virtual address
.endm

.macro senduart,rd,rx
str \rd, [\rx, #0x20] @ UTXH
```

```
.endm

.macro waituart,rd,rx
.endm

.macro busyuart,rd,rx
1001: ldr \rd, [\rx, #0x10] @ read UTRSTAT
tst \rd, #1 << 2 @ TX_EMPTY ?
beq 1001b
.endm
```

i.修改 arch/arm/kernel/setup.c 文件

此文件中的 setup_arch 非常关键，用来完成与体系结构相关的初始化：

```
void __init setup_arch(char **cmdline_p)
{
struct tag *tags = NULL;
struct machine_desc *mdesc;
char *from = default_command_line;

ROOT_DEV = MKDEV(0, 255);

setup_processor();
mdesc = setup_machine(machine_arch_type);
machine_name = mdesc->name;

if (mdesc->soft_reboot)
reboot_setup("s");

if (mdesc->param_offset)
tags = phys_to_virt(mdesc->param_offset);

/*
* Do the machine-specific fixups before we parse the
* parameters or tags.
*/
if (mdesc->fixup)
mdesc->fixup(mdesc, (struct param_struct *)tags,
&from, &meminfo);

/*
* If we have the old style parameters, convert them to
* a tag list before.
```

```
*/
if (tags && tags->hdr.tag != ATAG_CORE)
convert_to_tag_list((struct param_struct *)tags,
meminfo.nr_banks == 0);

if (tags && tags->hdr.tag == ATAG_CORE)
parse_tags(tags);

if (meminfo.nr_banks == 0) {
meminfo.nr_banks = 1;
meminfo.bank[0].start = PHYS_OFFSET;
meminfo.bank[0].size = MEM_SIZE;
}

init_mm.start_code = (unsigned long) &_text;
init_mm.end_code = (unsigned long) &_etext;
init_mm.end_data = (unsigned long) &_edata;
init_mm.brk = (unsigned long) &_end;

memcpy(saved_command_line, from, COMMAND_LINE_SIZE);
saved_command_line[COMMAND_LINE_SIZE-1] = '\0';
parse_cmdline(&meminfo, cmdline_p, from);
bootmem_init(&meminfo);
paging_init(&meminfo, mdesc);
request_standard_resources(&meminfo, mdesc);

/*
* Set up various architecture-specific pointers
*/
init_arch_irq = mdesc->init_irq;

#ifdef CONFIG_VT
#if defined(CONFIG_VGA_CONSOLE)
conswitchp = &vga_con;
#elif defined(CONFIG_DUMMY_CONSOLE)
conswitchp = &dummy_con;
#endif
#endif
}
```

j.修改 arch/arm/mm/mm-armv.c 文件（arch/arm/mm/目录中的文件完成与 ARM 相关的 MMU 处理）

修改

```
init_maps->bufferable = 0;
```

为

```
init_maps->bufferable = 1;
```

http://dev.yesky.com/153/2527653_4.shtml