华清远见 FARSIGHT 嵌入式培训专家

嵌入式学院 华清远见旗下品牌

构建嵌入式LINUX系统

易松华

2008-11-19

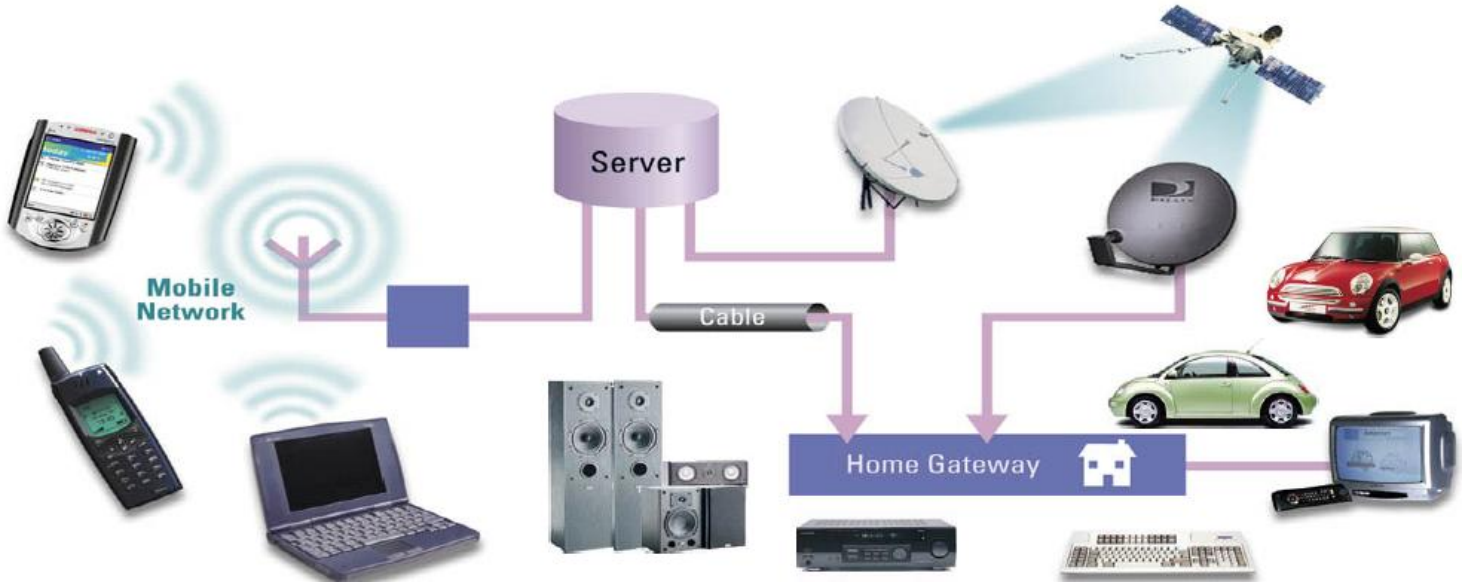
什么是嵌入式系统

- 嵌入式系统是以应用为中心，以计算机技术为基础，并且软硬件可裁剪，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统。





嵌入式系统的广泛应用



无线	机顶盒	汽车
手机	家庭网关	游戏
PDA	互联网	视频

嵌入式系统特点

- 嵌入式系统是将先进的计算机技术、半导体技术和电子技术与各个行业的具体应用相结合后的产物。
- 技术密集、资金密集、高度分散、不断创新的知识集成系统。

微机原理 **数字电路** **程序设计** **接口技术** **电路原理** **编译原理**

嵌入式系统特点

- 嵌入式系统面向特定应用，完成单个或一组联系紧密的功能
- 嵌入式系统产品功能特定，较通用计算机系统封闭，具有较长的生命周期
- 低功耗、体积小、集成度高，有限的存储器
- 嵌入式系统的硬件和软件都必须高效率地设计，量体裁衣
- 嵌入式系统本身不具备自主开发能力
- 需要配套开发工具和环境
- 一般有性能和实时性要求

实时系统特点

- 实时系统特点

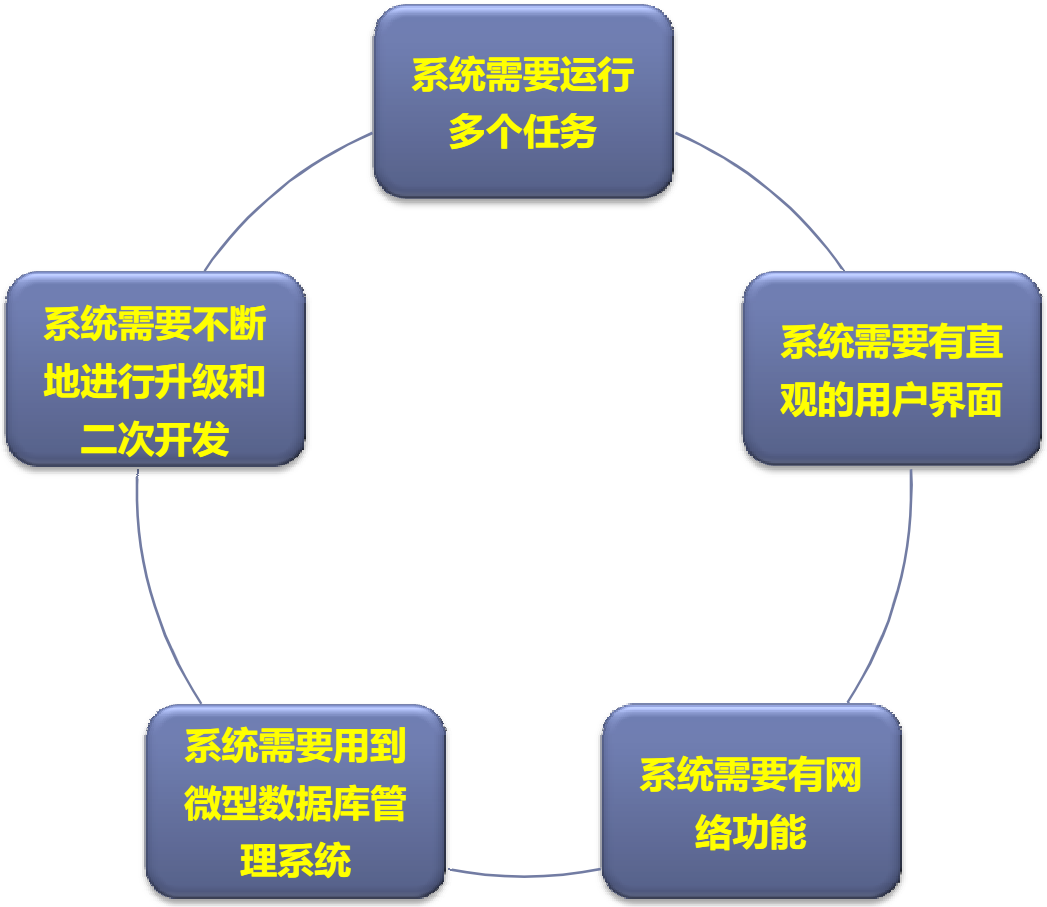
- 指规定的时限内必须完成规定的操作
  - 并非指速度快慢
  - 硬实时：超过时限完成任务会导致导致灾难性后果
  - 软实时：超过时限完成对任务会带来系统性能的严重下降

- 是否真正需要实时？

嵌入式技术的紧迫需求

- 精简、健壮的内核
- 友好、漂亮的GUI
- 安全、快速的网络通讯接口
- 功耗、性能的矛盾与平衡
- 轻巧、分布式的数据库相关技术

需要“嵌入式操作系统”？

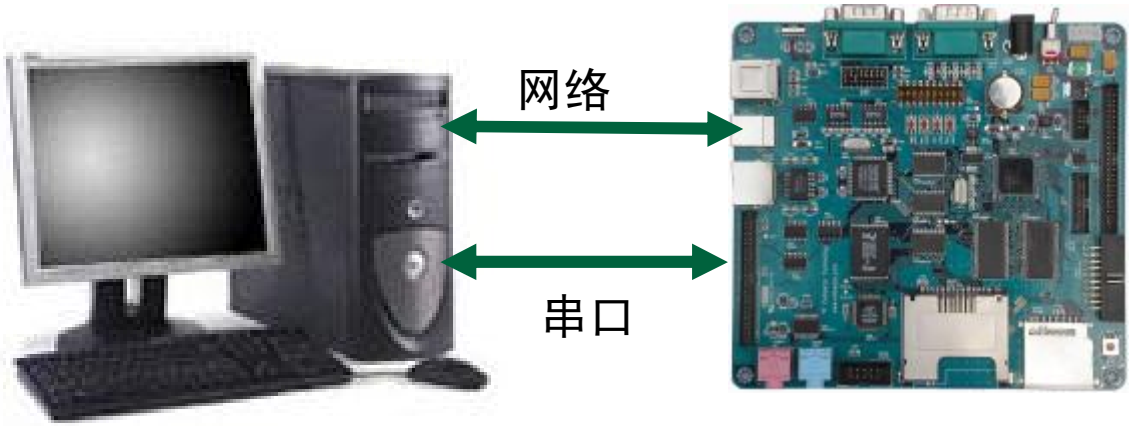

系统需要运行多个任务
系统需要有直观的用户界面
系统需要有网络功能
系统需要用到微型数据库管理系统
系统需要不断地进行升级和二次开发

嵌入式主控芯片百家争鸣

典型嵌入式操作系统

- 微内核（Microkernel kernel）
  - MicroC/OS-II
  - VxWorks
  - Threadx
  - *Nucleus*
  - FreeRTOS
- 宏内核（ Monolithic kernel）
  - Linux/ucLinux
  - Windows CE
  - Symbian

嵌入式开发与PC开发的比较



主机（**X86**）　　目标板（一般非**X86**）

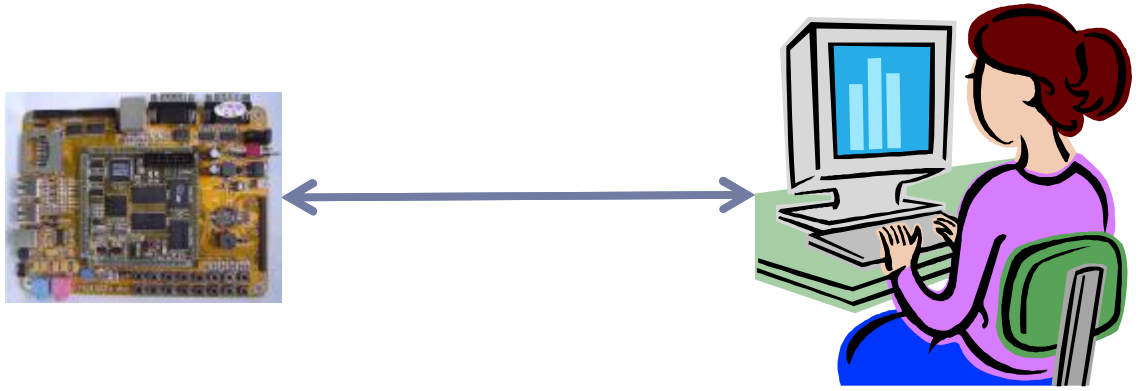
PC开发：本地开发本地运行

Self-Host/Self-Target

嵌入式开发：PC编译，目标板运行

Cross-Target Development

典型嵌入式开发



目标机	软件	宿主机
X	开发工具	√
X	源代码	√
√	调试器	√
√	操作系统	√
√	可执行程序	X

ARM LINUX的编译工具

- 开发主机
  - RedHat,Ubuntu,Windows（不推荐）
- 典型交叉编译工具
  - ELDK
  - CodeSourcery http://www.codesourcery.com/
  - 商业工具链
    - Montavista(可集成到图形开发环境DevRocket)
  - DIY
    - Crosstools
    - Buildroot

      经常使用的是uclibc

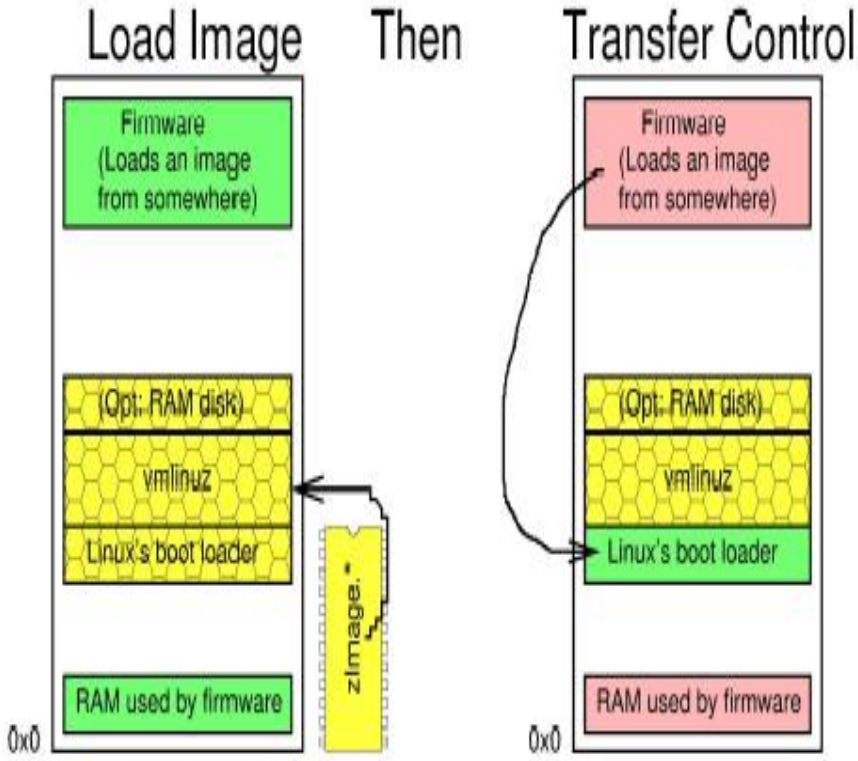
  IDE tools
    - Eclipse

ARM LINUX系统引导过程

- 预备知识
  - Bootloader
    - Uboot,RedBoot, VIVI等
  - 内核zImage/uImage
    - 内核加载地址和入口点(uImage only)
    - 内核压缩镜像vmlinuz
    - 内核解压缩代码
    - Ramfs根文件系统
      - 可选
      - 减少主机文件系统依赖，适合加速开发前期
  - 网络协议NFS/DHCP/TFTP

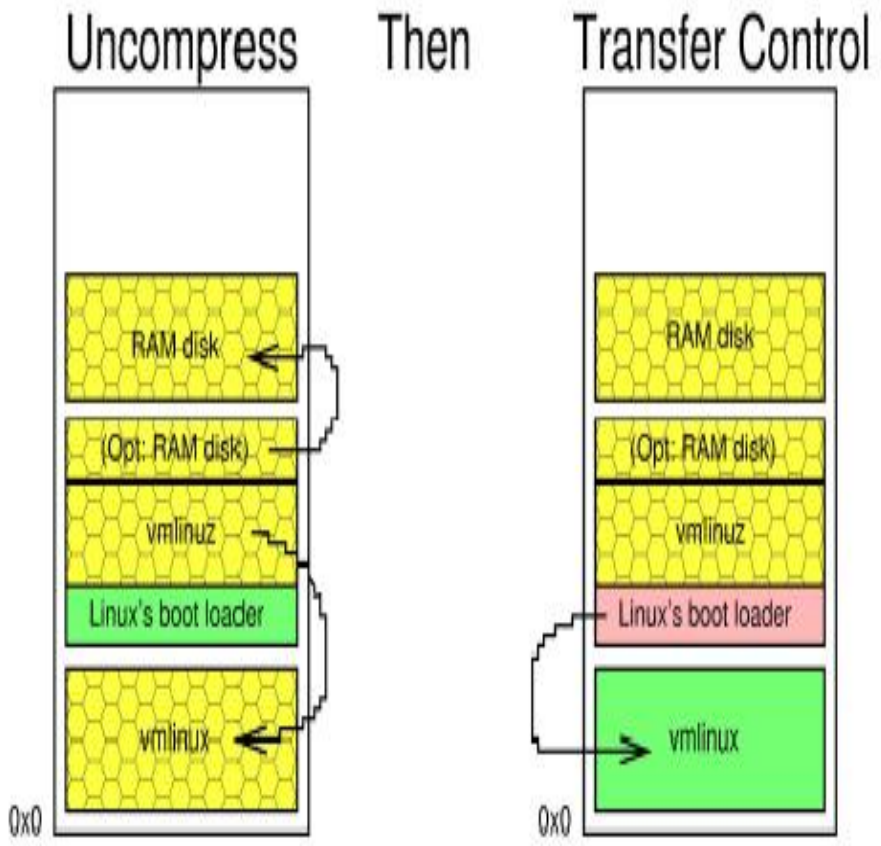
ARM LINUX系统引导过程

- Bootloader
  - 初始化基本运行环境
  - 加载内核到RAM中
  - 跳转到内核解压缩代码
  - 其他辅助功能



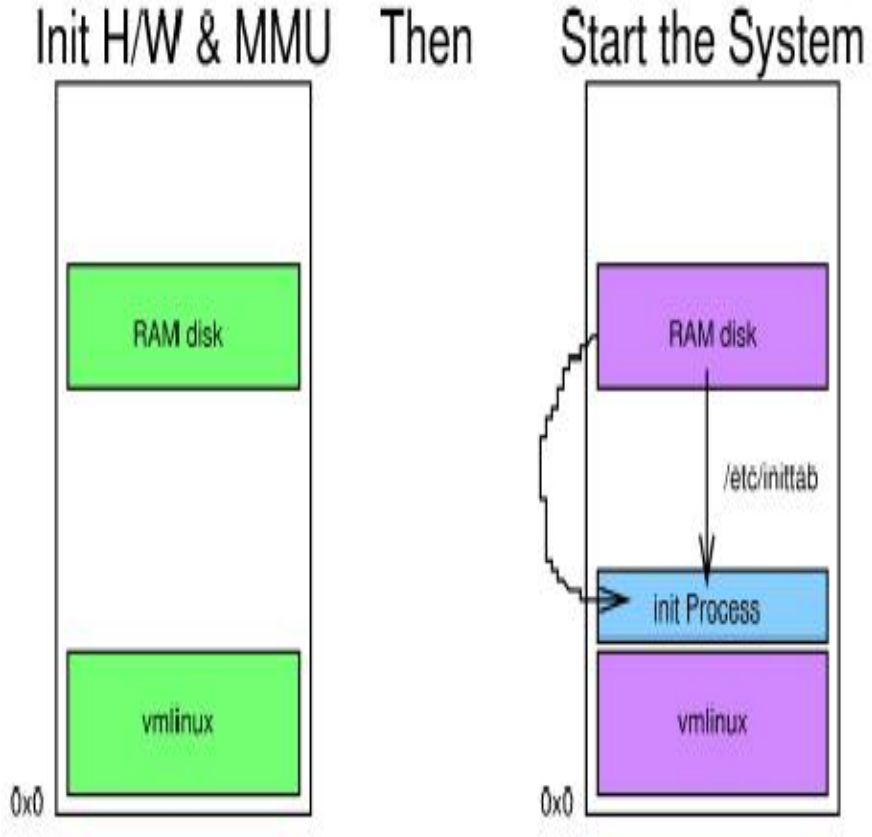
ARM LINUX系统引导过程

- 内核解压缩
  - 回收bootloader使用资源
  - 解压缩Vmlinuz
  - 解压缩可选的RAMDISK
  - 将控制权交给内核

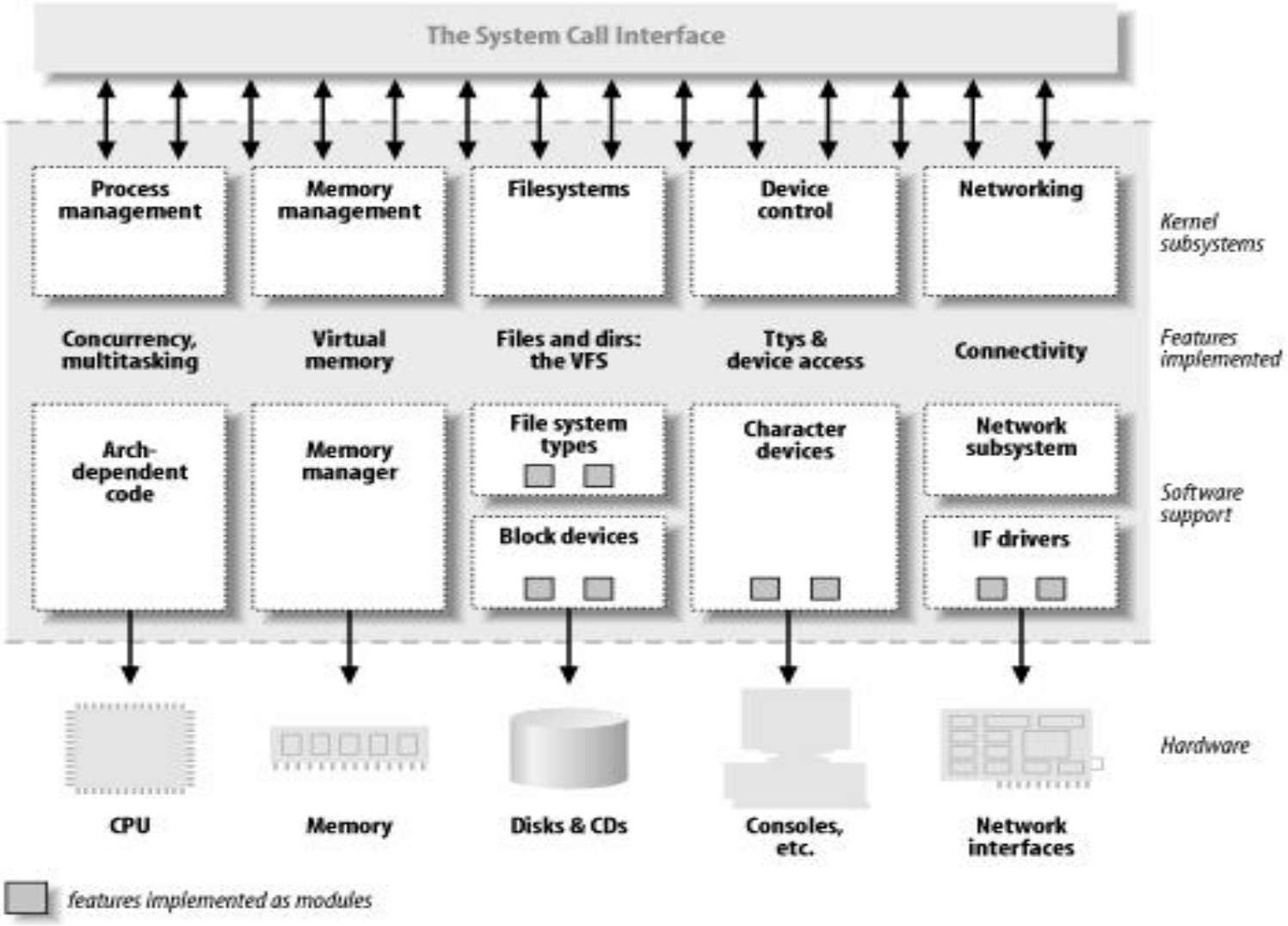


ARM LINUX系统引导过程

- 内核执行
  - 内核解压缩代码资源回收
  - 挂载根文件系统
    - 开发时用NFS（or ramdisk）
    - 产品用cramfs/jffs2等
  - 执行init
- 执行应用程序



LINUX组成结构图


The System Call Interface
Process management
Memory management
Filesystems
Device control
Networking
Kernel subsystems
Concurrency, multitasking
Virtual memory
Files and dirs: the VFS
Ttys & device access
Connectivity
Features implemented
Arch-dependent code
Memory manager
File system types
Block devices
Character devices
Network subsystem
IF drivers
Software support
CPU
Memory
Disks & CDs
Consoles, etc.
Network interfaces
Hardware
features implemented as modules

ARM LINUX 2.6移植步骤

- 准备开发板和相关配件
  - 要清楚自己的开发板的配置（有那些外围设备等）
- 下载内核源码、取得相关工具软件
  - Kernel，gcc，jtag
- 选定参考板
  - 选择一个内核中已经支持，且和自己的开发部相似的参考板为原型，进修修改
- 修改开发板相关代码
  - 完成BSP的修改移植
  - 如：arch/arm/mach-s3c2410/ 目录
- 添加驱动
  - 先添加基本驱动代码 （如：网卡）
- 编译、调试、下载内核

板级支持相关代码

- 移植相关目录和代码
  - arch/arm/mach-s3c2410
  - arch/arm/plat-s3c24xx
  - arch/arm/tools/mach-types
  - arch/arm/Kconfig
  - arch/arm/Makefile
  - arch/arm/boot/compressed
  - Include/asm-arm/arch-s3c2410
  - Makefile
- 添加基本驱动
  - 闪存驱动（MTD NAND）
  - 串口用来打印信息
    - 可以在内核启动之前打印信息
    - 内核启动后可以通过console打印信息
  - 网络支持可以使得开发调试更为方便
    - 如挂载网络文件系统 nfs

移植样例

- 体系结构: ARM920T
- 处理器: Samsung S3C2410
- 开发板: uCdragon FS2410
- Linux内核版本：2.6.22

移植相关

- 相关代码和目录
  - arch/arm/Kconfig
  - arch/arm/Makefile
  - arch/arm/boot/Makefile
  - arch/arm/mach-s3c2410/*
  - arch/arm/plat-s3c24xx/*
  - include/asm-arm/arch-s3c2410/*
  - arch/arm/tools/mach-types
  - arch/arm/boot/compressed/head.S
  - arch/arm/boot/compressed/Makefile
  - Makefile

Smdk2410基本信息定义

- arch/arm/mach-s3c2410/mach-smdk2410.c
  - 定义开发板描述信息

```
MACHINE_START(SMDK2410, "SMDK2410")
/* Maintainer: Jonas Dietsche */
.phys_io      = S3C2410_PA_UART,
.io_pg_offst= (((u32)S3C24XX_VA_UART) >> 18) & 0xfffc,
.boot_params          = S3C2410_SDRAM_PA + 0x100,
.map_io               = smdk2410_map_io,
.init_irq      = s3c24xx_init_irq,
.init_machine         = smdk2410_init,
.timer                = &s3c24xx_timer,
MACHINE_END
```

  - MACHINE_START 定义

include/asm-arm/mach/arch.h

struct machine_desc “__mach_desc_##_type” is defined

Smdk2410静态io空间映射（1）

- arch/arm/mach-s3c2410/mach-smdk2410.c

```
static struct map_desc smdk2410_iodesc[] __initdata = {
  /* nothing here yet */
};
static void __init smdk2410_map_io(void)
{
    s3c24xx_init_io(smdk2410_iodesc,
ARRAY_SIZE(smdk2410_iodesc));
    s3c24xx_init_clocks(0);
    s3c24xx_init_uarts(smdk2410_uartcfgs,
ARRAY_SIZE(smdk2410_uartcfgs));
}
```

Smdk2410静态io空间映射（2）

- arch/arm/mach-s3c2410/s3c2410.c
  - static struct map_desc s3c2410_iodesc[] __initdata = {
  - IODESC_ENT(CLKPWR),
  - IODESC_ENT(TIMER),
  - IODESC_ENT(WATCHDOG),
  - };
  - void __init s3c2410_map_io(struct map_desc *mach_desc, int mach_size)
  - {
  - /* register our io-tables */
  - iotable_init(s3c2410_iodesc, ARRAY_SIZE(s3c2410_iodesc));
  - iotable_init(mach_desc, mach_size);
  - }

Smdk2410静态io空间映射（3）

- arch/arm/plat-s3c24xx/cpu.c
  - static struct map_desc s3c_iodesc[] __initdata = {
  - IODESC_ENT(GPIO),
  - IODESC_ENT(IRQ),
  - IODESC_ENT(MEMCTRL),
  - IODESC_ENT(UART)
  - };
  - void __init s3c24xx_init_io(struct map_desc *mach_desc, int size)
  - {
    - .....
  - }

Nand 平台设备定义

- arch/arm/plat-s3c24xx/devs.c

```
static struct resource s3c_nand_resource[] = {
    [0] = {
        .start = S3C2410_PA_NAND,
        .end   = S3C2410_PA_NAND + S3C24XX_SZ_NAND - 1,
        .flags = IORESOURCE_MEM,        }
};
struct platform_device s3c_device_nand = {
    .name                   = "s3c2410-nand",
    .id             = -1,
    .num_resources          = ARRAY_SIZE(s3c_nand_resource),
    .resource       = s3c_nand_resource,
};
EXPORT_SYMBOL(s3c_device_nand);
```

platform_device结构

include/linux/platform_device.h

```
#ifndef _PLATFORM_DEVICE_H_
#define _PLATFORM_DEVICE_H_

#include <linux/device.h>

struct platform_device {
    const char      * name;
    u32             id;
    struct device   dev;
    u32             num_resources;
    struct resource             * resource;
};
```

Nand flash 分区定义(1)

- arch/arm/plat-s3c24xx/common-smdk.c
- static struct mtd_partition smdk_default_nand_part[] = {

```
    [0] = {
        .name           = "Boot Agent",
        .size           = SZ_16K,
        .offset         = 0,
    },
    [1] = {
        .name           = "S3C2410 flash partition 1",
        .offset = 0,
        .size           = SZ_2M,
    },
    [2] = {
        .name           = "S3C2410 flash partition 2",
        .offset = SZ_4M,
        .size           = SZ_4M,
    },
    …
    [7] = {
        .name           = "S3C2410 flash partition 7",
        .offset = SZ_1M * 48,
        .size           = SZ_16M,
    }};
```

Nand flash 分区定义(2)

```
static struct s3c2410_nand_set smdk_nand_sets[] = {
    [0] = {
        .name                   = "NAND",
        .nr_chips    = 1,
       .nr_partitions= ARRAY_SIZE(smdk_default_nand_part),
       .partitions    = smdk_default_nand_part,
    },
};/* choose a set of timings which should suit most 512Mbit * chips and beyond.*/
static struct s3c2410_platform_nand smdk_nand_info =
{
    .tacls                  = 20,
    .twrph0                 = 60,
    .twrph1                 = 20,
    .nr_sets        = ARRAY_SIZE(smdk_nand_sets),
    .sets           = smdk_nand_sets,
};
```

平台相关代码分析-common-smdk2410.c

- arch\arm\plat-s3c24xx\common-smdk.c

```
static struct platform_device __initdata *smdk_devs[] = {
    &s3c_device_nand,
    &smdk_led4,
    &smdk_led5,
    &smdk_led6,
    &smdk_led7,
};
```

```
static struct s3c24xx_board smdk2410_board
__initdata = {
.devices        = smdk2410_devices,
.devices_count =
ARRAY_SIZE(smdk2410_devices)
};
```

平台设备注册

arch/arm/plat-s3c24xx/common-smdk.c

```
void __init smdk_machine_init(void){
/* Configure the LEDs (even if we have no LED support)*/
    s3c2410_gpio_cfgpin(S3C2410_GPF4, S3C2410_GPF4_OUTP);
    s3c2410_gpio_cfgpin(S3C2410_GPF5, S3C2410_GPF5_OUTP);
    s3c2410_gpio_cfgpin(S3C2410_GPF6, S3C2410_GPF6_OUTP);
    s3c2410_gpio_cfgpin(S3C2410_GPF7, S3C2410_GPF7_OUTP);
    s3c2410_gpio_setpin(S3C2410_GPF4, 1);
    s3c2410_gpio_setpin(S3C2410_GPF5, 1);
    s3c2410_gpio_setpin(S3C2410_GPF6, 1);
    s3c2410_gpio_setpin(S3C2410_GPF7, 1);
     if (machine_is_smdk2443())
         smdk_nand_info.twrph0 = 50;
     s3c_device_nand.dev.platform_data = &smdk_nand_info;
     platform_add_devices(smdk_devs, ARRAY_SIZE(smdk_devs));
    s3c2410_pm_init();
}
```

内核启动参数

- Boot options >
  - Default kernel command string:
    - console=ttySAC0,115200 noinitrd root=/dev/nfs rw ip=bootp
    - 另外一个：
    - console=ttyS0,115200 =/dev/mtdblock/2 rw init=/linuxrc ip=192.168.0.118:192.168.0.99:192.168.0.1:255.255.255.0:farsight:eth0:off display=dh240

修改Makefile(1)

- 顶层Makefile
  - Makefile

```
ARCH=arm
CROSS_COMPILER=arm-linux-
```

修改Makefile(2)

- arch/arm/Makefile
  - machine-$(CONFIG_ARCH_S3C2410) := s3c2410
  - ifneq ($(machine-y),)
  - MACHINE := arch/arm/mach-$(machine-y)/
  - else
  - MACHINE :=
  - Endif
  - textofs-y := 0x00008000
  - TEXT_OFFSET := $(textofs-y)
- **arch/arm/mach-s3c2410/Makefile.boot**
  - zreladdr-y := 0x30008000
  - params_phys-y := 0x30000100
- arch/arm/boot/Makefile
  - ZRELADDR := $(zreladdr-y)
  - PARAMS_PHYS := $(params_phys-y)
  - INITRD_PHYS := $(initrd_phys-y)export ZRELADDR INITRD_PHYS PARAMS_PHYS

修改Makefile(3)

- **arch/arm/boot/compressed/Makefile**
  - ifeq ($(CONFIG_ZBOOT_ROM),y)
  - ZTEXTADDR := $(CONFIG_ZBOOT_ROM_TEXT)
  - ZBSSADDR := $(CONFIG_ZBOOT_ROM_BSS)
  - else
  - ZTEXTADDR := 0
  - ZBSSADDR := ALIGN(4)
  - endif
  - ifneq ($(INITRD_PHYS),)
  - LDFLAGS_vmlinux += --defsym initrd_phys=$(INITRD_PHYS)
  - endif
  - ifneq ($(PARAMS_PHYS),)
  - LDFLAGS_vmlinux += --defsym params_phys=$(PARAMS_PHYS)
  - endif
- Include/asm-arm/memory.h
  - #ifndef PAGE_OFFSET
  - #define PAGE_OFFSET UL(0xc0000000)
  - #endif
- 关于地址的定义参考 **Documentation/arm/Porting**

配置、编译

- make menuconfig
- make zImage（or make uImage)
- make moudles

文件系统

- Linux严重依赖文件系统
  - 设备, 数据, 视频等
- 系统所使用的存储设备决定文件系统的大小和类型
- 为了高可靠性，使用日志文件系统
  - 在事件失败时候自动执行文件recovery

文件系统(1)

- 文件系统支持 (只列出部分特性)
  - CramFS
    - A compressed, read-only file system
    - File sizes are limited to less than 16MB, and the maximum file system size is a little over 256MB.
    - Useful for root file systems
      - Applications and libraries do not change at run time
    - Typically, better compression and faster performance than you see from JFFS2
  - SquashFS
    - A compressed, read-only file system
    - Better compression than cramfs
    - Better runtime performance

文件系统(2)

- 文件系统支持 (只列出部分特性)
  - JFFS/JFFS2
    - A journaling file system that places itself directly on flash chips with the disk emulation
    - Automatically performs wear-leveling and garbage collection
    - The file system is compressed allowing for more storage in the same amount of flash
  - Yet Another Flash File System (YAFFS2)
    - Designed and optimized for NAND Flash chips
    - Larger Flash sizes and faster erase and write processes
    - Journaling support that automatically provides automatic wear-leveling and power failure recovery
  - 更多细节请参考华清远见的“嵌入式系统班”讲义

嵌入式Linux应用开发班

- 嵌入式Linux开发环境构建(应用层)
- 嵌入式Linux进程及进程间通讯开发
- 嵌入式Linux下的网络开发
- 嵌入式GUI与数据库开发

嵌入式Linux系统开发班

- 嵌入式Linux开发环境构建(系统层)
- 开发Linux系统引导程序(bootloader)
- 配置编译Linux内核
- 移植Linux内核源码
- 调试Linux内核方法
- 集成部署Linux系统(rootfs)

www.farsight.com.cn

Thanks !