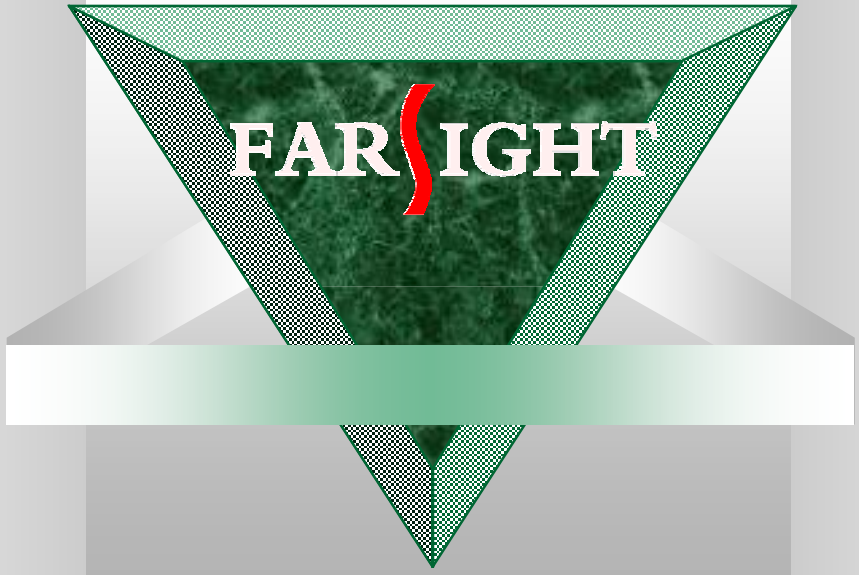


基于Linux的驱动开发

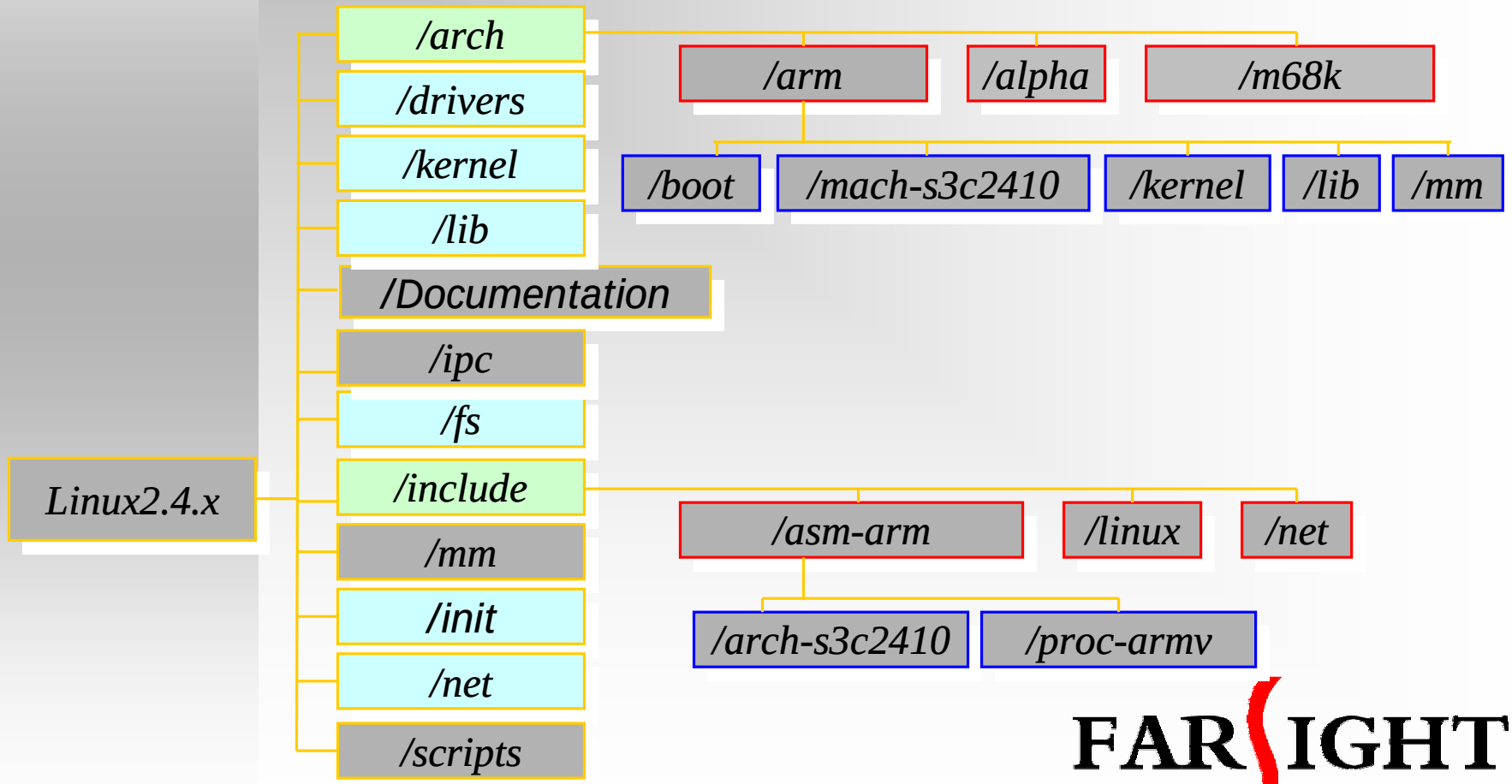
刘淼

Linux内核与C代码

- Linux内核庞大，结构复杂
  - 对Linux 2.4内核的统计：1万个文件，4百万行代码
  - 对Linux 2.6内核的统计：1.5万个文件，6百万行代码
- Linux内核的主体使用GNU C，在ANSI C上进行了扩充
  - Linux内核必须由gcc编译编译
  - gcc和Linux内核版本并行发展，对于版本的依赖性强
  - Linux 2.6内核建议使用gcc 3.3以上版本，C99编程风格
- 内核代码中使用的一些编程技巧，在通常的应用程序中很少遇到
- 学好Linux、首先要学好C语言

FARSIGHT

linux 2.4 的内核目录结构



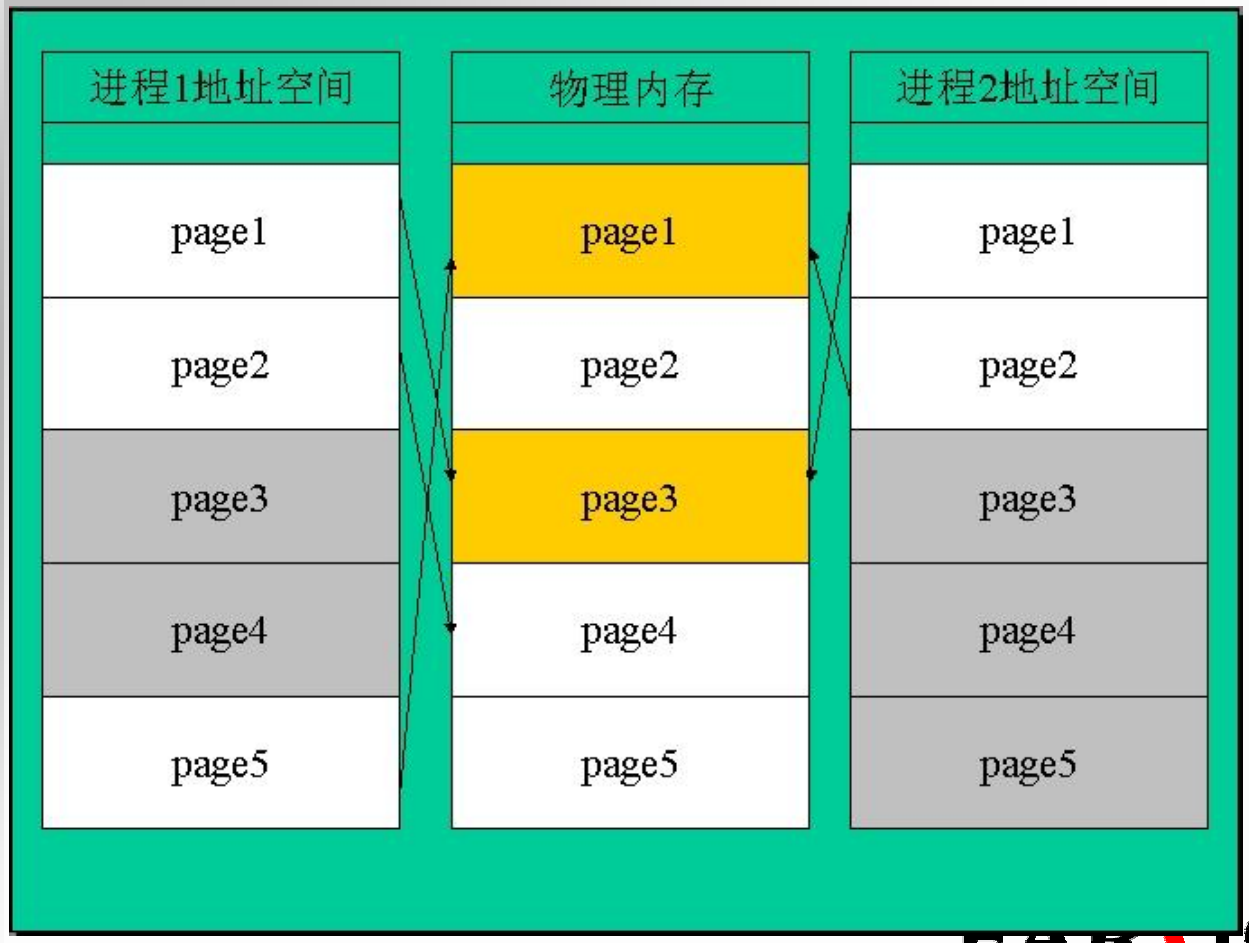
FARSIGHT

Linux的驱动程序

- Linux下对外设的访问只能通过驱动程序。uClinux下可以在应用层直接访问外设，操作寄存器口，但是无法处理中断、DMA、抢占、原子操作等——不推荐使用
- Linux对于驱动程序有统一的接口，以文件的形式定义系统的驱动程序：
  - Open、Release、read、write、ioctl…
- 驱动程序是内核的一部分，可以使用中断、DMA等操作
- 驱动程序需要在用户态和内核态之间传递数据
- 对于复杂的应用可以考虑是用mmap

FARSIGHT

什么是MMU
（Memory management unit）



Linux下设备和模块的分类

按照上述系统内核的功能，Linux中把系统的设备定义成如下三类：

- 字符设备
- 块设备
- 网络设备

FARSIGHT

Linux下的设备

- Linux的设备以文件的形式存在于/dev目录下
- 设备文件是特殊文件，使用ls /dev -l命令可以看到：

```
crw-------  1 root root   10,   7 Aug 31  2002 amigamouse1
crw-------  1 root root   10, 134 Aug 31  2002 apm_bios
brw-rw----  1 root disk   29,   0 Aug 31  2002 aztcd
```

FARSIGHT

主设备号和次设备号

- 主设备号标识设备对应的驱动程序
- 一个驱动程序可以控制若干个设备，次设备号提供了一种区分它们的方法
- 系统增加一个驱动程序就要赋予它一个主设备号。这一赋值过程在驱动程序的初始化过程中

int register_chrdev(unsigned int major, const char*name,struct file_operations *fops);

FARSIGHT

使用devfs

- 在Linux 2.4的内核里引入了devfs来解决
  linux下设备文件管理的问题
- 在驱动程序中通过devfs_register()函数创建设备文件系统的节点
- 系统启动的时候mount设备文件系统
- 所有需要的设备节点都由内核自动管理。
  /dev目录下只有挂载的设备

FARSIGHT

Linux 2.6内核与devfs

- Linux 2.6内核引入了sysfs文件系统为每个系统的硬件树进行分级处理
- Devfs在Linux 2.6中被标记为舍弃的特性（在Linux 2.6.15及以后的版本则取消了对它的支持），而使用udev。
  - 维护动态设备
  - 从sysfs获得的信息，可以提供对特定设备的固定设备名。对于热插拔的设备，这尤其重要
  - udev 是在用户空间的脚本文件，这很容易被编辑和修改
  - 可以代替hotplug脚本
- 为了保证旧应用程序的兼容性，在嵌入式系统中，是用devfs还是一个好方法。即使在Linux 2.6.15内核以后，也可以通过ndevfs（nano devfs）补丁提供对devfs特性的兼容

FARSIGHT

在Linux 2.6内核中使用udev

- 建议，在2.6.15以后的版本中使用udev
- 使用ramfs作为udev的载体
  - mount –t ramfs none /dev
- udev使用的规则集位于/etc/udev/*
- udev的官方地址：
  http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html
- 参考文章：Writing udev rules

FARSIGHT

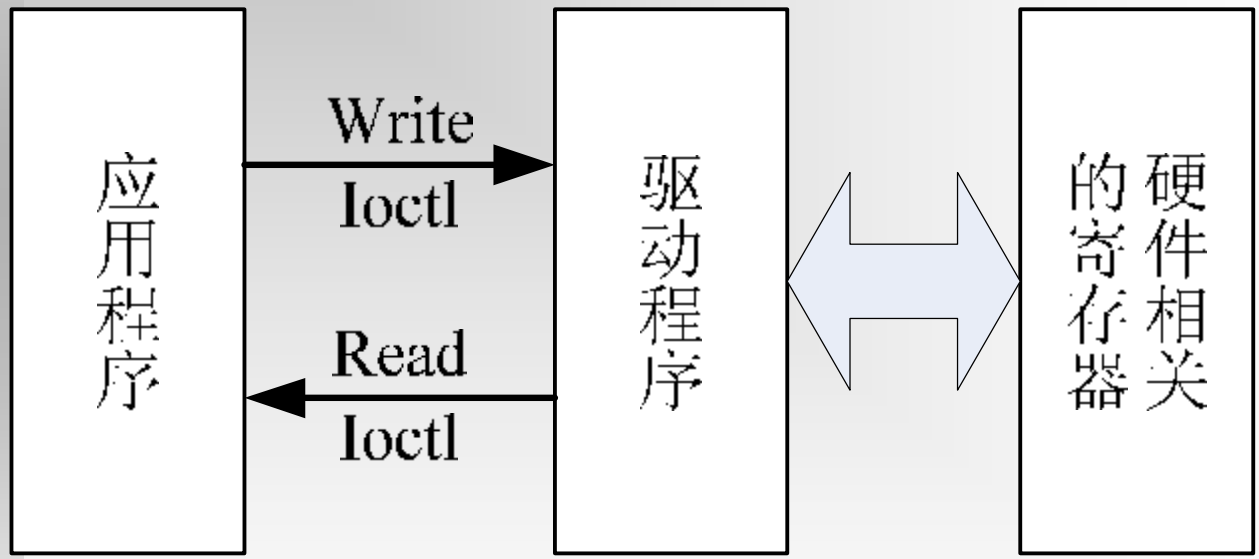
设备驱动程序的使用与测试方法

- 应用层使用open、close、read、write系统调用——需要编写应用程序

- 使用系统命令可以进行最基本的测试：
  - cat /dev/urandom
  - echo /dev/urandom > /dev/fb0
  - dd if=/dev/touchscreen of=/var/tmp/test bs=16 count=100

FARSIGHT

一个简单的Linux驱动程序原理

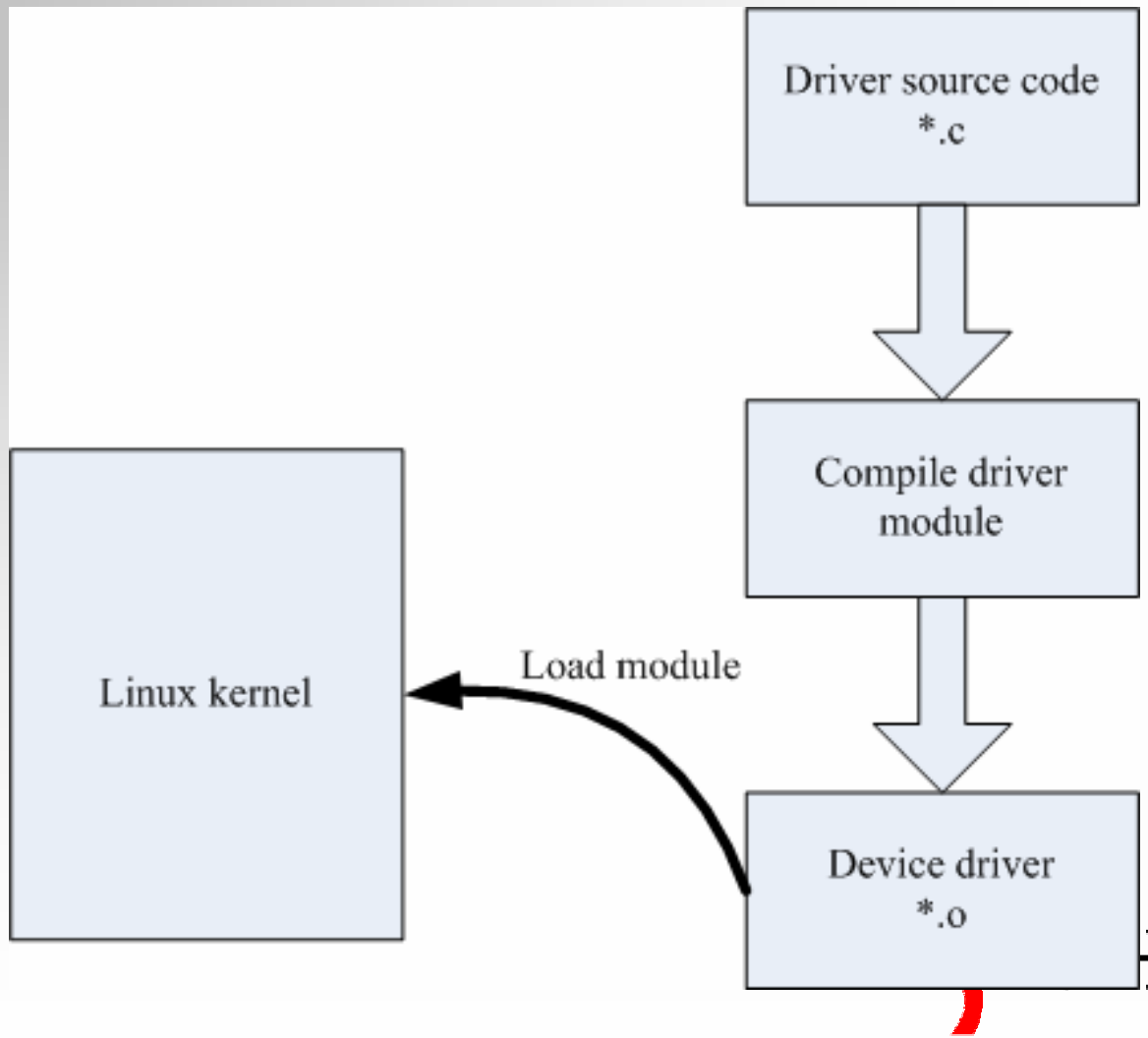


Linux驱动程序加载方式

- 驱动程序直接编译入内核
  - 驱动程序在内核启动时就已经在内存中
  - 可以保留专用存储器空间
- 驱动程序以模块形式存储在文件系统里，需要时动态载入内核
  - 驱动程序按需加载，不用时节省内存
  - 驱动程序相对独立于内核，升级灵活

FARSIGHT

Linux驱动程序模块加载



嵌入式Linux下常见的文件系统

- RomFS：只读文件系统，可以放在ROM空间，也可以在系统的RAM中，嵌入式linux中常用来作根文件系统
- RamFS：利用VFS自身结构而形成的内存文件系统，使用系统的RAM空间
- JFFS/JFFS2：为Flash设计的日志文件系统
- Yaffs：专门为Nand Flash设计
- proc：为内核和内核模块将信息发送给进程提供一种机制，可以查看系统模块装载的信息
- devFS：设备文件系统

FARSIGHT

Linux上的Ext2fs

- 支持 4 TB 存储、文件名称最长1012 字符
- 可选择逻辑块
- 快速符号链接

- Ext2不适合flash设备
- 是为象 IDE 设备那样的块设备设计的，逻辑块大小必须是 512 byte、1 KB、2KB等
- 没有提供对基于扇区的擦除 / 写操作的良好管理
  - 如果在一个扇区中擦除单个字节，必须将整个扇区复制到 RAM，然后擦除，再重写入
- 在出现电源故障时，Ext2fs 是不能防止崩溃的
- 文件系统不支持损耗平衡，缩短了flash的寿命

FARSIGHT

jffs/jffs2文件系统的优缺点

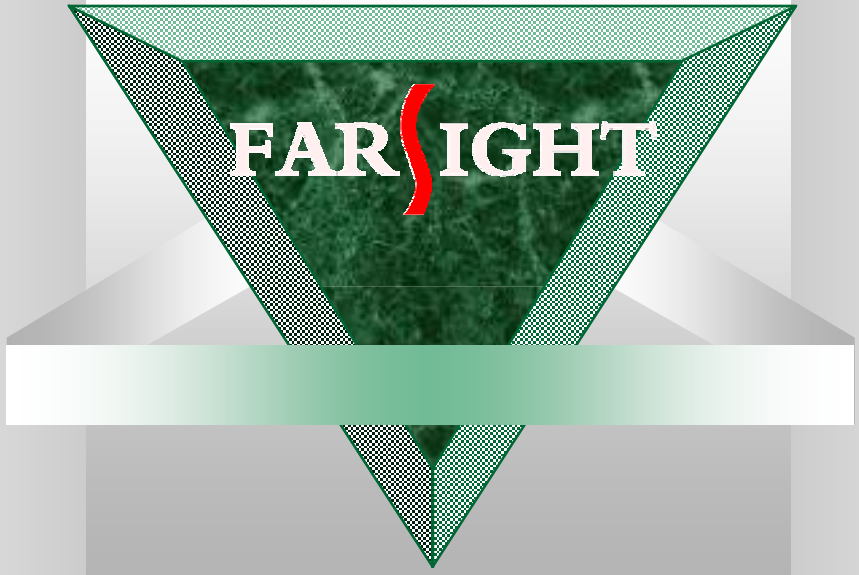
- 日志文件系统
- 提供了更好的崩溃、掉电安全保护
- jffs2支持对flash的均匀磨损
- 在扇区级别上执行闪存擦除 / 写 / 读操作要比 Ext2文件系统好
- 文件系统接近满时，JFFS2 会大大放慢运行速度——垃圾收集

FARSIGHT

Nand上yaffs文件系统的优势

- 专门为Nand flash设计的日志文件系统
- jffs／jffs2不适合大容量的Nand flash
  - jffs的日志通过jffs_node建立在RAM中，占用RAM空间：对于128MB的Nand大概需要4MB的空间来维护节点
  - 启动的时候需要扫描日志节点，不适合大容量的Nand flash
- FAT系统没有日志

FARSIGHT



谢谢！