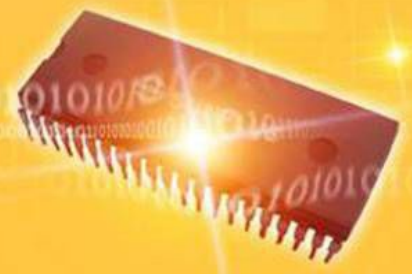
嵌入式系统工程师



Linux内核开发移植

大纲

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux内核开发

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux内核版本变迁及其获得

- Linux是最受欢迎的自由电脑操作系统内核，是一个用C语言写成，符合POSIX标准的类Unix操作系统
- 诞生于1991年10月5日，由芬兰黑客Linus Torvalds为尝试在英特尔x86架构上提供自由免费的类Unix操作系统而开发的
- Linux操作系统的诞生、发展、和成长过程依赖于五个重要支柱：unix操作系统、minix操作系统、GNU计划、POSIX标准和互联网

Linux内核版本变迁及其获得

- UNIX操作系统：
  - 是美国贝尔实验室的ken Thompson(肯.汤普逊)和Dennis Ritchie(丹尼斯.里奇)于1969年夏在DEC PDP-7小型计算机上开发的一个分时操作系统，后经Dennie Ritchie于1972年用C语言进行改写，使得unix得到了极大的发展,是Linux操作系统一直模仿和超越的对象
- MINIX操作系统：
  - 是由荷兰阿姆斯特丹自由大学计算机科学系的塔能鲍姆教授(Andrew S.Tanenbaum)在1987年根据unix开放源代码编写，主要用于学生学习操作系统原理，没有任何的实际应用,可以说是Linux的直接父亲

Linux内核版本变迁及其获得

- GNU计划和自由软件基金会(Free Software Foundation)：
  - 由Richard M.Stallman于1984年创办，旨在开发一个免费、类似unix的操作系统—GNU系统及其开发工具(如Emacs编辑系统、BASH shell程序、GCC、GDB等)，后来与Linux内核结合成为了现在的GNU/Linux

- POSIX（Portable Operating System Interface)
  - 可移植操作系统接口,由电气和电子工程师协会（IEEE）开发，用来统一unix、Linux各分支编程接口，以提高其通用性和可移植性，使得Linux的发展结束了初期的混乱发展阶段，进入了一个新的时期

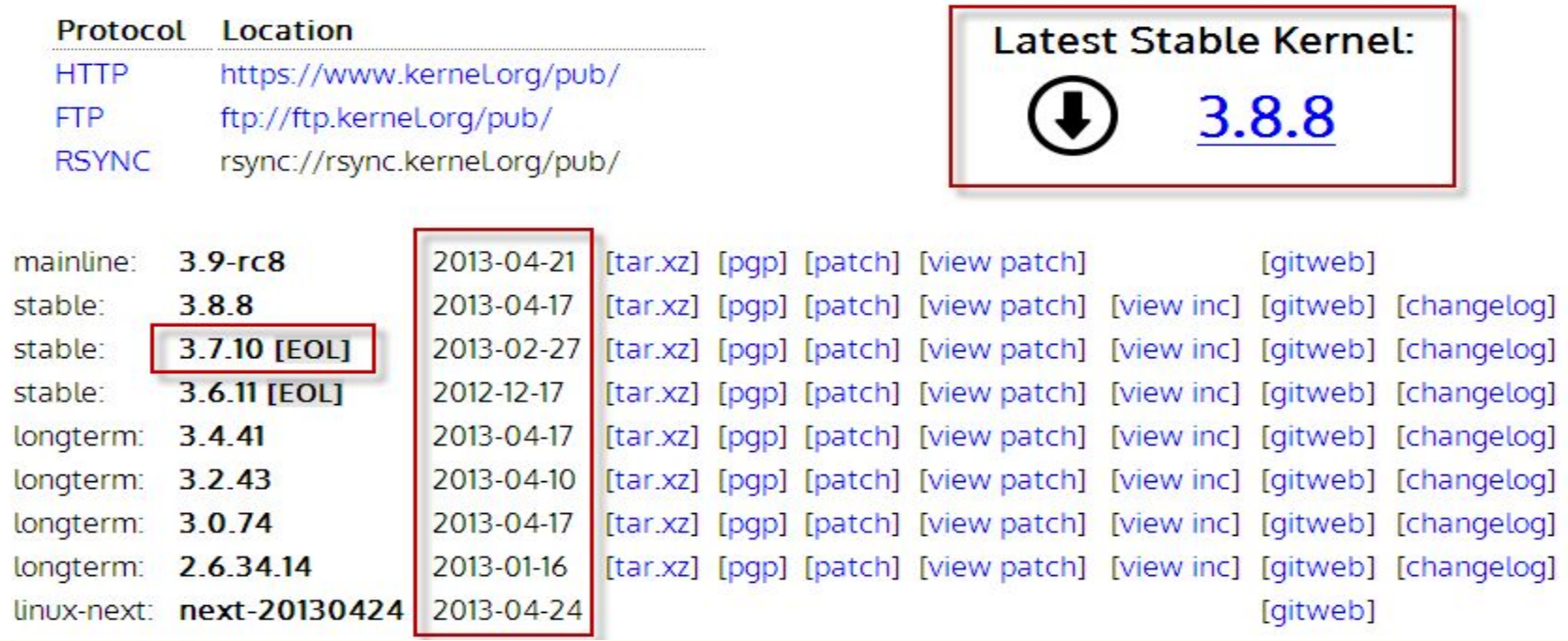
Linux内核版本变迁及其获得

- Linus在unix与MS-DOS的夹缝中，在minix的引导下，在GNU、POSIX的帮助下于1991年8月发布了Linux内核的第一个版本0.01版，造就了后来几十年的辉煌：

1991年08月，大约有10000行代码的Linux v0.01版
1991年10月，Linus Torvalds发布了Linux v0.02，
成为一个独立的操作系统的诞生
1993年，由上百名程序员参与，发布版本Linux v0.99
1994年03月，Linux v1.0.0，共有17万行代码（第一个正式版）
1995年03月，Linux v1.2.0，约30万行代码，支持多平台
1996年06月，Linux v2.0.0，约40万行代码，支持多处理器
1999年01月，Linux v2.2.0，约180万行代码
2001年01月，Linux v2.4.0，约330万行代码
2003年12月，Linux v2.6.0，约600万行代码
2009年06月，2.6.30的内核，约1160万行代码
2011年07月，3.0的内核，约1300万行代码

Linux内核版本变迁及其获得

- http://www.kernel.org是内核源码的主要来源，所有来自全世界的对Linux源码的修改最终都会汇总到这个网站，由Linus领导的开源社区对其进行鉴别和修改最终决定是否进入到Linux主线内核源码中，以下是www.kernel.org主页截图



Protocol	Location
HTTP	https://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 3.8.8

mainline:	3.9-rc8	2013-04-21	[tar.xz]	[pgp]	[patch]	[view patch]		[gitweb]
stable:	3.8.8	2013-04-17	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
stable:	3.7.10 [EOL]	2013-02-27	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
stable:	3.6.11 [EOL]	2012-12-17	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
longterm:	3.4.41	2013-04-17	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
longterm:	3.2.43	2013-04-10	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
longterm:	3.0.74	2013-04-17	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
longterm:	2.6.34.14	2013-01-16	[tar.xz]	[pgp]	[patch]	[view patch]	[view inc]	[gitweb] [changelog]
linux-next:	next-20130424	2013-04-24						[gitweb]

Linux内核版本变迁及其获得

- 内核版本号从Linux1.0以后主要分为两个阶段：
  - Linux1.0-2.6，数字包括四部分“A.B.C.D”
    - A代表主版本号，如1994年的1.0，1996年的2.0，2011年的3.0
    - B代表次版本号，表示一些重大的修改，偶数表示稳定版，奇数表示开发版
    - C代表末版本号，代表着一个新版本的发布(包括安全补丁、bug修复、新增功能、或驱动程序)，一般数字变化范围比较大
    - D代表一些BUG修复，对已经加入的安全补丁、bug修复、新增功能或驱动程序做的微调或添加新的特性，2.6版本之后经常出现

Linux内核版本变迁及其获得

- Linux2.6之后的版本，数字主要包括三部分"A.B.C"或"A.B.C-rcn"
  - A代表主版本号
  - B代表次版本号，随着一个新版本的发布而增加，与上一阶段中的数字"C"功能类似，但不再代表稳定与否
  - C代表稳定版本号，如果只有数字则代表稳定版本，如果带"-rcn"表示最终稳定版本之前的候选版本(Release Candidate)，n代表候选版本号
- 各版本之间的关系及命名规则可以参考网站：http://en.wikipedia.org/wiki/Linux_kernel#Version_numbering
- 稳定的内核具有工业级的强度，可以广泛地应用和部署

Linux内核版本变迁及其获得

- 注意：page9中的EOL：end of life：指从此往后不在对应的版本上进行更新。下面展示了Linux内核的版本变迁过程：

Linux内核开发

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux内核结构

- Linux内核是Linux系统软件的核心，它的性能对整个系统的性能起决定作用
- Linux内核文件数目将近2万多个，他们分别位于顶层目录下的17个子目录中，了解Linux内核的工作过程，必须从了解其目录结构做起
- 我们后面研究并移植的内核为:3.0.8，首先我们从官网上下载3.0.8的内核源码**Linux-3.0.8.tar.bz**，将其在虚拟机上解压，使用tree命令可以在虚拟机下查看其目录结构

Linux内核结构

- Linux内核源代码主要包含以下子目录：
- **arch**

  与体系结构相关的代码。对应于每个支持的体系结构，有一个相应的子目录如x86、arm等与之对应，相应目录下有对应的芯片与之对应
- **drivers**

  设备驱动代码，占整个内核代码量的一半以上，里面的每个子目录对应一类驱动程序，如：

  char:字符设备、block:块设备、net:网络设备等
- **fs**

  文件系统代码，每个支持的文件系统有相应的子目录，如cramfs，yaffs，jffs2等

Linux内核结构

- include

  这里包括编译内核所需的大部分头文件
  - 与平台无关的头文件放在include/linux子目录下
  - 各类驱动或功能部件的头文件（/media、/mtd、/net等）
  - 与体系相关的头文件

    arch/arm/include/
  - 与平台相关的头文件路径

    arch/arm/mach-s5pv210/include/mach
- lib

  与体系结构无关的内核库代码

  与体系结构相关的内核库代码在arch/arm/lib下

Linux内核结构

- **init**

  内核初始化代码,其中的main.c中的start_kernel函数是系统引导起来后运行的第1个函数，这是研究内核开始工作的起点

- **ipc**

  内核进程间通信代码

- **mm**

  与体系结构无关的内存管理代码

  与处理器体系结构相关的代码在arch/arm/mm下

Linux内核结构

- kernel

  内核管理的核心代码

- net

  网络部分代码，其每个目录对应于网络的一个方面

- scripts

  存放一些脚本文件，如配置内核时用到的make menuconfig命令

- Documentation

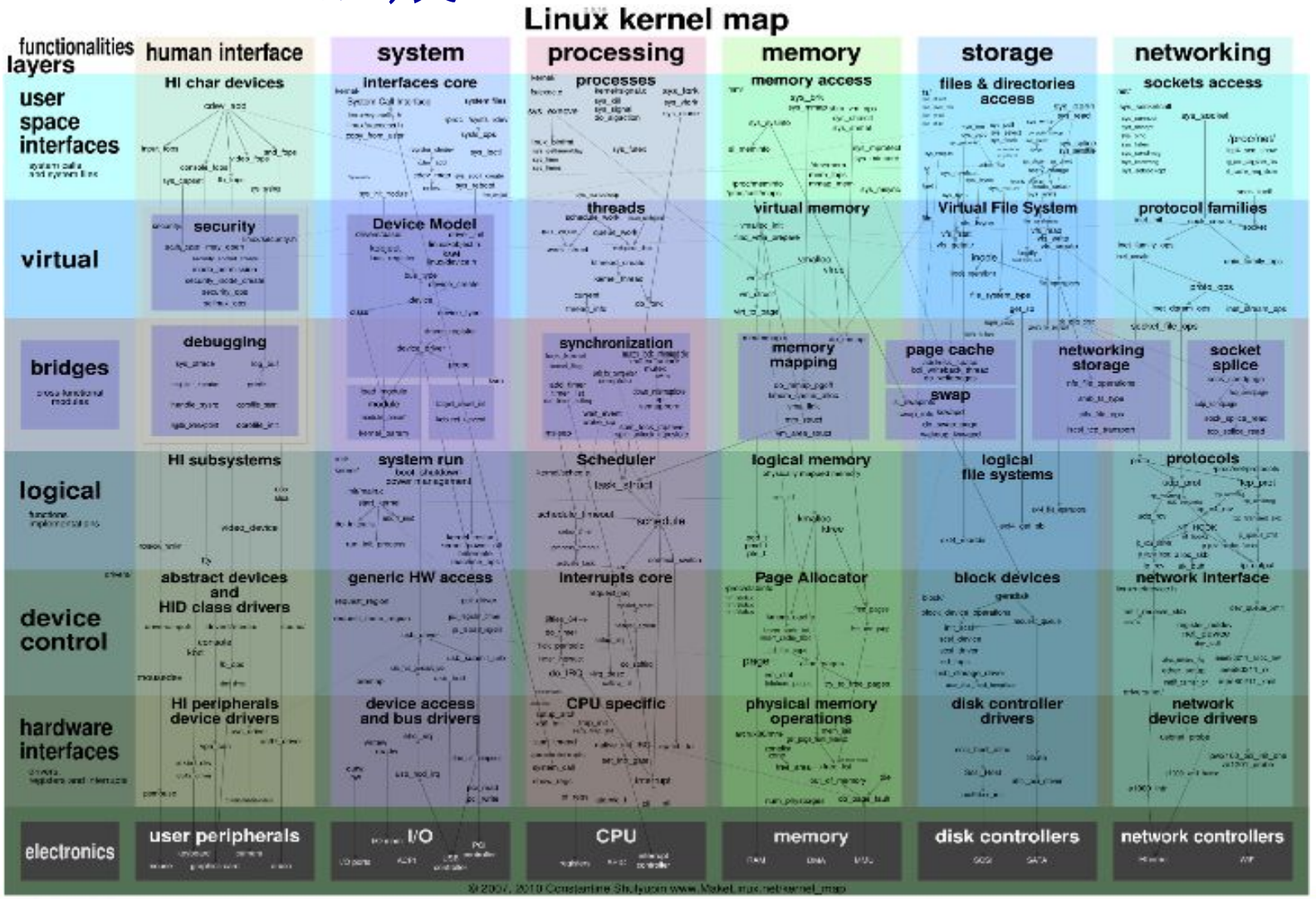
  内核相关文档

- crypto

  常用加密及散列算法，和一些压缩及CRC校验算法

Linux内核结构

- Linux kernel组成：



Linux内核结构

功能层	人机接口	系统	进程	内存	存储系统	网络
用户空间接口 与系统调用等	字符设备	设备驱动 核心接口	进程控制	内存访问接口	目录和文件 的访问	套节子访问
虚拟层	安全模块 (SELinux)	设备模型 可变的设备层	内核线程操作	逻辑上连续 的虚拟内存	虚拟文件系统	协议族接口
交叉功能模块 ——桥梁作用	系统调试接口	设备模型 不常变的驱动层	同步机制	内存映射	页缓冲及 网络存储	套节子功能接口
逻辑功能实现	人机交互子系统 (如tty设备)	内核启动 运行相关	调度机制	物理上连续 的虚拟内存	逻辑文件系统 (如FAT)	传输层、网络层 各种协议
设备控制	抽象类和 人机类设备	通用硬件访问	中断核心层	页分配器	块设备	网络设备接口
硬件接口驱动	外围人机设备 驱动	设备访问与 总线驱动	CPU体系结构 相关	物理内存操作 接口	磁盘控制器 驱动	网络设备驱动
电气层	人机外围设备	I/O接口资源	CPU资源	物理内存单元	物理磁盘	网络设备控制器

Linux内核结构

- Linux内核部分模块：
  - 进程管理、进程调度、系统调用、中断处理、下半部和工作队列、内核同步、定时管理、内存管理、虚拟文件系统、块设备、网络子系统、进程地址空间、模块机制等等...

Linux内核结构

- Linux编程风格
  - 像所有其他大型软件项目一样，Linux制定了一套编码风格
  - 跟选择一个唯一确定的风格相比，到底选择什么样的风格反而显得不是那么重要
  - 编码风格的主要规范伴随着Linus一贯的幽默，都记录在内核源码的Documentation/Coding Style中了。
  - 制定编码风格的目的，是为了提高编程效率，吸引更多的开发者眼球。

Linux内核结构

- 缩进
  - 采用制表符(Tab)进行缩进，而不是空格
  - 禁止制表符和空格混合使用

- 花括号使用如下：

```
if (flag) {
    fun();
    ……
}
```

```
void
fun (void)
{
}
```

```
if (flag)
    fun();
```

Linux内核结构

- 命名规范
  - Linux规定名称中不允许使用混合大小写字符
  - 单词之间用下划线分隔
  - 避免取有疑惑的简单名称，如pad()，应该写成platform_add_devices()
- 代码长度
  - 每行尽量不超过80个字符(可进行有意义分行切割)
  - 函数体代码长度尽量不超过两屏
  - 函数局部变量尽量不超过十个
  - 根据函数使用频率和函数体大小可以使用inline声明，以提高调用效率

Linux内核结构

- 注释
  - 一般情况注释用于描述代码可以做什么和为什么要做，尽量不写实现方式
  - 函数的修改和维护日志统一集中在文件开头
- 在程序中对ifdef的处理
  - 一般不这么用：

```
……
#ifdef  CONFIG_FUN
        fun();
#endif
……
```

Linux内核结构

- 而是在声明处使用ifdef
  - 比如：

```
#ifdef  CONFIG_FUN
void fun (void)
{
        ……
}
#else
inline void fun (void) { }
#endif
```

Linux内核结构

- 其他
  - 指针中的“*”号应靠近变量名，而不是类型关键字
  - 函数之间用空行隔开
  - 函数导出申明紧跟在函数定义的下面
  - 等等……
- 代码风格的事后修正
  - indent命令是大多数Linux系统中都带有的工具，当得到一段与内核编码风格大相径庭的代码时，可以通过这个工具进行调整：

```
#indent -kr -i8 -ts8 -sob -l80 -ss -bs -psl  <filename>
```

Linux内核开发

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux内核的启动引导过程

- 在了解Linux启动流程之前，首先需要了解Linux镜像的格式及其产生过程

Linux内核的启动引导过程

- Linux内核有多种镜像格式：
  - **Image**：直接生成并未压缩的内核，一般用于PC机
  - **zImage**：Image的压缩版，采用gzip进行压缩，比Image体积小，但启动时需要进行自解压，嵌入式系统中一般采用此种方法
  - **uImage**：是u-boot专用的一种内核镜像格式，它是在zImage的基础上又添加了一个长度为0x40的标签头，在u-boot启动时会去掉此头信息，仍按zImage启动，头信息主要用于区分不同格式的内核镜像
  - **xipImage**：片上执行的未压缩内核，（如norflash等）
  - **bootpImage**：将内核与根文件系统制作在一起的镜像

Linux内核的启动引导过程

- zImage镜像产生过程：

  vmlinux->Image->compressed/vmlinux->zImage

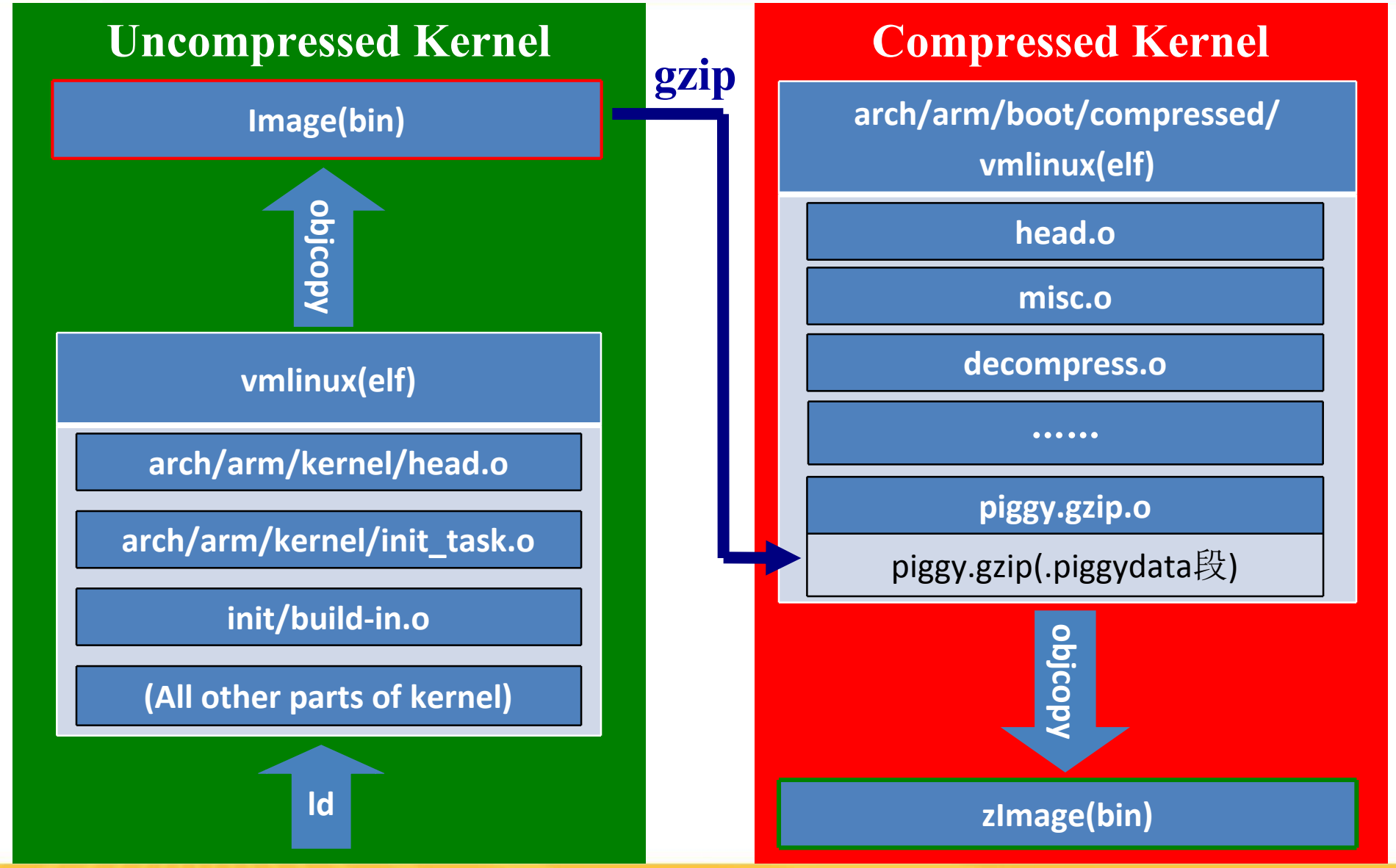
1、**vmlinux**是由以下内核代码生成的非压缩镜像(arch/arm/kernel/head.s、usr/、kernel/、mm/、fs/、ipc/、security/、crypto/、lib/、drivers/、net/等等)

2、**Image**是使用objcopy工具对vmlinux进行二进制化处理得到的镜像

3、**arch/arm/boot/compressed/vmlinux**由压缩的Image和compressed/head.S、misc.c等文件组成

4、**zImage**是由compressed/vmlinux使用objcopy工具二进制化得到

Linux内核的启动引导过程



Linux内核的启动引导过程

- **Linux内核的启动过程大体上可以分为3个阶段：**

1、内核解压（汇编+C）

- 主要由arch/arm/boot/compressed/对zImage完成解压，并调用call_kernel跳转到下阶段代码

2、板级引导阶段（汇编）

- 主要进行cpu和体系结构的检查、cpu本身的初始化以及页表的建立等

3、通用内核启动阶段（C语言）

- 进入init/main.c文件，从start_kernel开始进行内核初始化工作，最后调用rest_init

Linux内核的启动引导过程

- rest_init()会创建内核第一个线程，并进入线程函数kernel_init()
- 在kernel_init()中会初始化各种驱动并建立起标准输入/标准输出/错误输出，最后调用init_post()
- 在init_post()中会释放初始化内存段，标志着内核启动完成，并努力寻找一个用户进程，通过kernel_execve()函数加载，将该进程作为系统的第一个用户进程init，进程号为1
- 内核启动完成，接下来就是用户的事儿了

Linux内核开发

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux内核的配置与编译

- Linux内核配置裁剪过程

  make menuconfig

- Linux内核编译过程
  - 根据配置裁剪的结果配合Makefile完成内核编译

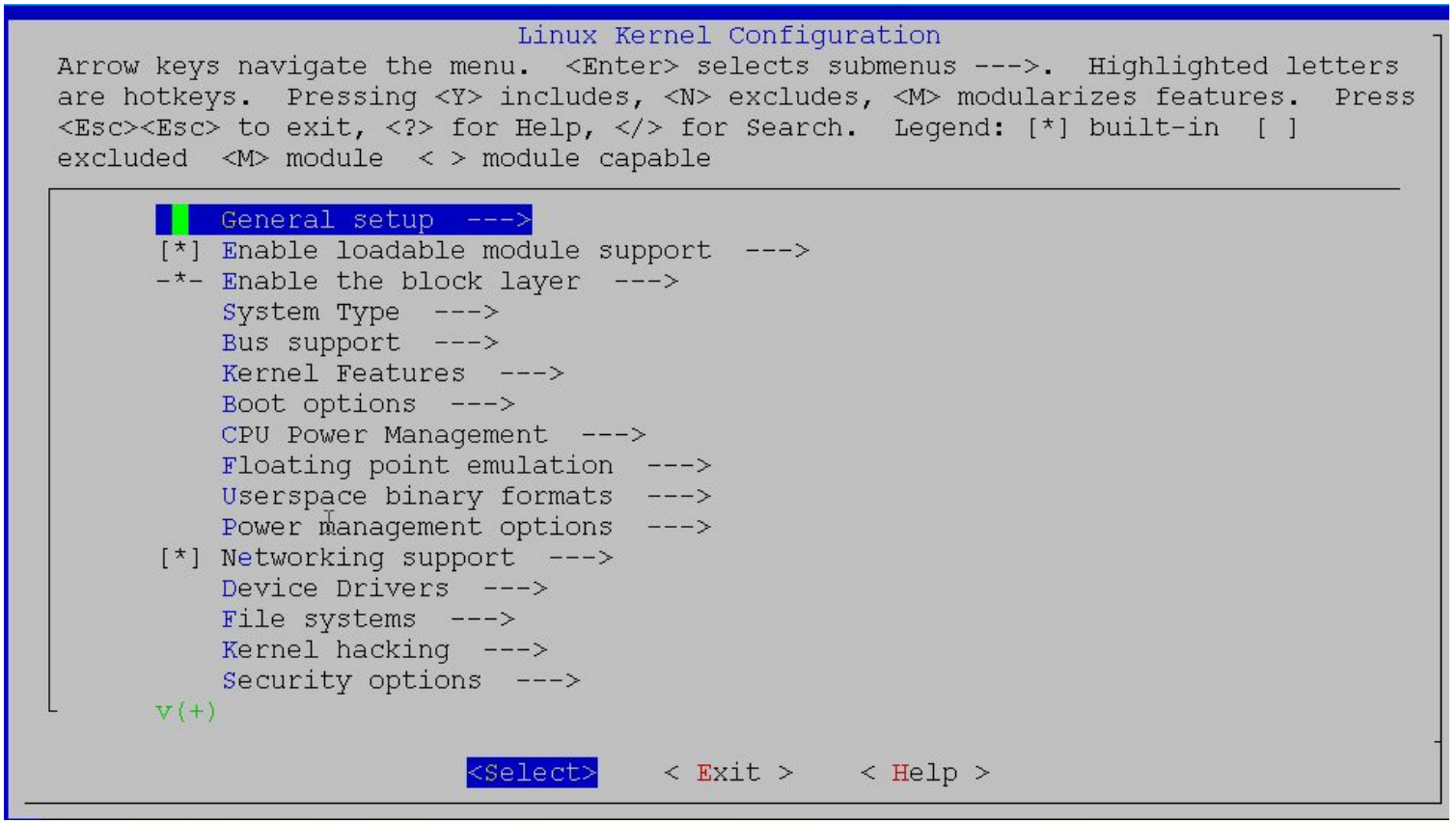
- 本节我们将讨论Linux内核这一配置编译机制
- 并通过驱动程序s5pv210-key15-simple.c(1*5键盘驱动)添加到内核源码来验证整个过程

Linux内核的配置与编译

- 在Linux2.6以后的版本中，文件的组织是通过Kconfig和Makefile来实现的
- 通过每层目录的Kconfig和Makefile实现了整个Linux内核的分布式配置
  - **Kconfig**：对应内核模块的配置菜单
  - **Makefile**：对应内核模块的编译选项
- 如果有新的内核源码需要加入，怎样通过make menuconfig对其进行裁剪？
- 当用户通过make menuconfig选中某内核模块时，怎样结合Makefile完成编译？

Linux内核的配置与编译

- 当执行make menuconfig时，配置工具会自动根据根目录下的ARCH变量读取arch/$(ARCH)/Kconfig文件来生成配置界面，这个文件是所有文件的总入口，其它目录的Kconfig都由它来引用，如图所示：



Linux内核的配置与编译

- 这里我们以ARM平台为例讲解其具体配置过程

  1、当执行make menuconfig时，系统首先读取arch/arm/Kconfig生成整个配置界面

  2、在读取配置界面的同时，系统会读取顶层目录下的.config文件生成所有配置选项的默认值

  3、当修改完配置并保存后，系统会更新顶层目录下的.config文件

  4、当执行make时，各层的Makefile会根据.config文件中的编译选项来决定哪些文件会被编译到内核中，或编译成模块

Linux内核的配置与编译

- Kconfig语法格式可以参考具体文件，如：drivers/char/Kconfig

```
#
# Character device configuration
#

menu "Character devices"

config VT
        bool "Virtual terminal" if EMBEDDED
        depends on !S390
        select INPUT
        default y
        ---help---
          If you say Y here, you will get support
```

Linux内核的配置与编译

- config代表一个选项的开始,最终会出现在.config中(会自动增加一个CONFIG_前缀)
- bool代表此选项仅能选中或不选中,bool后面的字符串代表此选项在make menuconfig中的名字
  tristate：代表可以选择编译、不编译、编译成模块
  string：字符串；hex：16进制的数；int：10进制的数
- depends on:依赖其余的选项
- default：默认选项值
- select：表示当前config被选中时，此选项也被选中
- menu/endmenu：表示生成一个菜单
- choice/endchoice：表示选择性的菜单条目
- comment：注释信息，菜单和.config文件中都会出现
- source：用于包含其它Kconfig

Linux内核的配置与编译

- 将自己开发的内核代码加入到Linux内核中，需要有3个步骤：
  1. 确定把自己开发代码放入到内核合适的位置
  2. 把自己开发的功能增加到Linux内核的配置选项中，使用户能够选择此功能
  3. 构建或修改Makefile，根据用户的选择，将相应的代码编译到最终生成的Linux内核中去
- 下面我们以把一个key1*5键盘驱动添加进内核为例讲解配置机制的具体应用
- 添加步骤如下：

Linux内核的配置与编译

Step1：将s5pv210-key15-simple.c拷到drivers/char/目录下

Step2: vi driver/char/Kconfig,在Kconfig文件结尾，在endmenu的前面加入一个config选项

```
619 config UNSP210_KEY15
620         bool "key1*5 driver for sunplusedu unsp210 boards"
621         default y
622         help
623           this is GPIO driver for sunplusedu unsp210 boards.
624
625 endmenu
```

Linux内核的配置与编译

Step3：make menuconfig(添加配置选项)

Device driver-→

character devices→

[*] key1*5 driver for sunplusedu unsp210 boards

Step4：vi driver/char/Makefile添加内容如下：

```
64 obj-$(CONFIG_UNSP210_KEY15)      += s5pv210-key15-simple.o
65 obj-$(CONFIG_JS_RTC)             += js-rtc.o
66 js-rtc-y = rtc.o
```

Step5：make（更新内核镜像到开发板）

Step6：交叉编译测试程序,并放到开发板运行

#arm-linux-gcc key15_test.c -o key15_test

Linux内核开发

- Linux内核版本变迁及其获得
- Linux内核结构
- Linux内核启动引导过程
- Linux内核的配置与编译
- Linux3.0.8内核移植

Linux3.0.8内核移植

- 以Linux3.0.8内核在开发板上的移植为例，学习如何从一个纯净的Linux内核编译、配置得到适合在一个特定平台运行的镜像。具体移植步骤如下：
  1、源码解压
  2、修改顶层Makefile
  3、修改链接地址
  4、修改机器ID号并拷贝默认配置文件
  5、yaffs文件系统的补丁支持
  6、nand flash驱动移植
  7、内核配置、编译
- 具体参考文档《Linux-3.0.8内核移植》文档

Linux3.0.8内核移植

- 任务：
  1、添加键盘驱动到Linux-2.6.35.7
  2、Linux-3.0.8内核移植
  3、Linux-3.0.8内核LCD移植
  4、Linux-3.0.8内核修改开机logo
  5、添加键盘驱动到Linux-3.0.8

凌阳教育官方微信：Sunplusedu

Tel: 400-705-9680 , BBS: www.51develop.net , QQ群：241275518