

宅学部落

专注嵌入式精品教程

www.zhaixue.cc

Linux内核编程：入门指南

王利涛

Linux内核开发领域

- 招聘职位(Title)
 - 初级/高级/资深Linux嵌入式工程师
 - 系统软件工程师、嵌入式驱动工程师
 - Linux内核工程师、Linux驱动工程师
 - Linux开发工程师、架构师

内核工程师 VS 驱动工程师

- 内核工程师

- 岗位分布

- 互联网、云计算、安全
 - OS发行商、芯片原厂

- 技能树

- 内存管理、调度、实时性
 - 进程、同步与并发
 - 存储、文件系统
 - 网络子系统
 - 虚拟化、区块链、容器

- 驱动工程师

- 岗位分布

- 芯片原厂、方案厂商
 - 嵌入式设备厂商

- 技能树

- 内核编译、移植
 - 设备模型、驱动框架
 - USB、PCI、I2C、UART
 - 芯片手册、硬件电路
 - 音视频编解码

驱动与内核的关系

- 驱动主要用来驱动各种硬件
 - 驱动是内核代码的一部分：内核中80%的代码都是驱动
 - 驱动不同的CPU、硬件平台、外设、热插拔设备
- 内核一般用来提供OS的各种服务
 - Linux内核划分：BSP、驱动、基础服务、应用服务
 - 基础服务：进程、调度、内存管理、同步机制、中断
 - 应用服务：虚拟化、容器、网络、存储、文件系统
 - 随着应用场景变化，新的功能会不断添加进来

学习驱动能找什么工作？

学习驱动能找什么工作？

- 驱动工程师的岗位分布
 - IP厂商、咨询公司：ARM、Linaro
 - 芯片原厂
 - 国际厂商：TI、CSR、NXP、英飞凌、高通、Freescale
 - 国内厂商：海思、联发科、瑞芯微、全志、君正、龙芯
 - AI芯片厂商、RISC-V
 - 方案厂商：模组、解决方案、智能硬件、开发板
 - 设备公司：智能音箱、无人机、智能冰箱、电视
 - 嵌入式外包公司

学习驱动能找什么工作？

- 驱动工程师的主要工作
 - IP解决方案：IP授权、配套的软件包
 - 使能硬件：硬件不断升级、添加新的feature
 - Bug fix：基于某个CPU硬件平台、修复各种Bug
 - 性能优化：释放硬件的最大性能
 - 系统集成：适配不同的CPU架构、硬件平台
 - 输出稳定、高性能、可适配的BSP软件包、Turkey解决方案

学习驱动能找什么工作？

- 驱动的分类及演化

- Misc驱动：一个驱动对应一个硬件、兼容性最差
- 字符驱动、设备驱动：读写方式
- 总线型驱动
 - 物理总线：USB、I2C、PCI
 - 虚拟总线：platform、device tree
- 驱动框架：framebuffer、input、DRM
- 用户态驱动：硬件抽象层HAL

学习内核能找什么工作？

学习内核能找什么工作？

- 内核工程师的岗位分布
 - 互联网大厂：服务器、云计算、分布式存储、网络
 - 安全厂商：安全加固、移动安全、物联网安全
 - 基础软件：Red Hat、SUSE、UOS、VMware
 - 芯片公司：Intel、AMD、IBM、华为、三星
 - 其他领域：超算、服务器、桌面PC

学习内核能找什么工作？

- 内核工程师的主要工作
 - 内核移植与架构：芯片原厂内核组
 - 业务模块的开发与优化
 - 各种Bug fix：云、容器、服务器、存储
 - 维护kernel稳定版本、各种patch的backporting
 - 内核安全加固
 - 开发新的功能：内核社区、Linux基金会、大厂
 - 内核参数调节、性能优化：运维、调试、服务器

为什么要学习Linux内核？

- 从事Linux内核开发的优势
 - 好就业：嵌入式、服务器、运维、云、安全
 - 待遇高：研发核心岗、技术要求高
 - 吃经验：底层调试、系统分析、全栈能力
 - 不内卷：门槛高、要求高、经验贬值小
- 从事Linux内核开发的劣势
 - 就业机会相对较少
 - 投入产出性价比低
 - 远离业务层

如何学习Linux内核？

Linux内核学习难点

- Linux内核的版本变化
 - Linux-0.01:
 - 源文件：88个文件、10239行代码
 - 支持平台：X86
 - Linux-5.4
 - 源文件：48815个文件、超过2500万行代码(.c/.h/.S)
 - 支持平台：X86、ARM、RISC-V、MIPS等20多个平台
 - Linux-5.10
 - 源文件：52333个文件、超过2700万行代码(.c/.h/.S)
 - 支持平台：X86、ARM、RISC-V、MIPS等20多个平台

Linux内核的学习难点

- 开发者人数、提交修改次数

2019年,平均每小时10.7次提交

年份	版本工具	提交(Commits)	开发人数
1991	-	-	1
2002	Bitkeeper	15,474	497
2005	Bitkeeper/Git	23,553	1372
2008	Git	48,851	2304
2011	Git	56,405	2806
2014	Git	75,650	3580
2017	Git	80,826	4175
2019	Git	82,371	4249

Linux内核的学习难点

- 数据结构

- 数据结构之间关系错综复杂、成员巨多
- 层层内嵌、相互指向、陷入代码的热带雨林中
- 侵入式(intrusive)数据结构、内嵌双向链表

- 函数实现

- 单个函数最大行数可达1000行以上
- 分支多：调用几十甚至上百个子函数： `start_kernel()`
- 层次深：层层封装、调用层深5级起步
- 不同版本的API接口变动

Linux内核的学习难点

- 跟踪代码容易断篇
 - 语法障碍：
 - GCC编译器的特性语法、复杂的宏定义
 - 底层编译链接机制：.section、Makefile、链接脚本、汇编
 - 底层初始化相关机制：xxx_init、__init、initcall机制
 - 找不到函数的定义、调用链跟踪失败
 - 大量的重名函数、不知道调用哪个、代码分析误入歧途
 - 一个接口多个实现：xxx_ops->read()
 - 函数指针调用、函数指针作为参数
 - 链表的添加、删除、内嵌、offset
 - 各个子系统之间相互交叉调用

小结

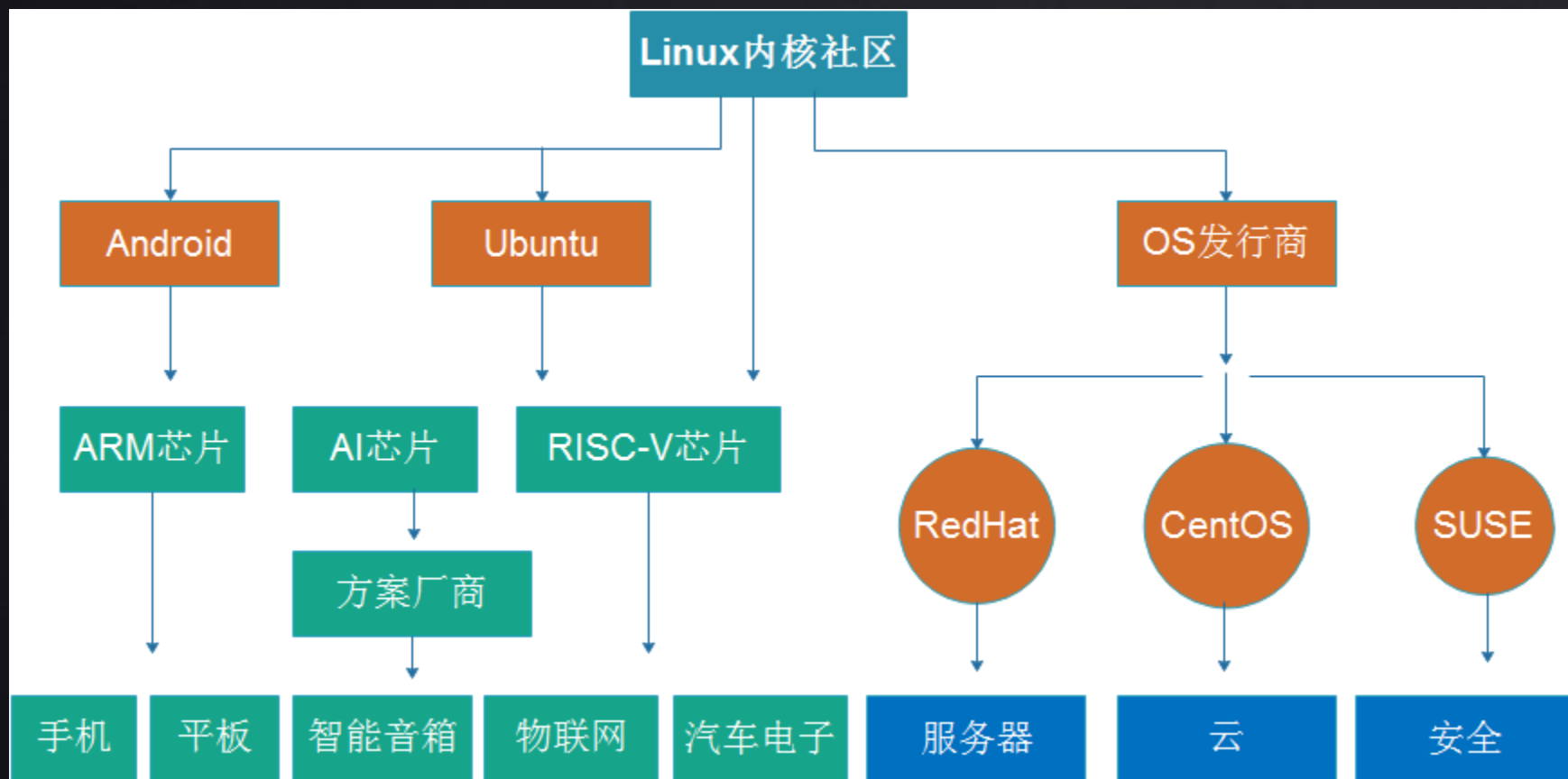
- 在宏观框架和细节把握上无法达到平衡
- 对内核的理解，到底要达到什么程度？
- 需求驱动、自身目标定位、以解决问题为导向
- 学习层次不同、需求不同，关注点就不一样

Linux内核驱动开发生态

2007~2019内核提交统计

组织、机构、公司	内核提交(Commits)	所占百分比
None	93,225	11.95%
Intel	78,068	10.01%
Red Hat	69,443	8.90%
Unknow	31,919	4.09%
IBM	29,358	3.79%
SUSE	27,239	3.49%
Linaro	24,740	3.17%
Consultant	23,081	2.96%
Google	21,779	2.79%
Samsung	20,160	2.58%
AMD	17,781	2.28%
...
Arm	7,646	0.98%
Linux Foundation	6,109	0.78%

Linux内核驱动开发生态



Linux内核驱动开发生态

- Linux基金会\内核社区
 - 2000年成立，非营利性联盟，致力于促进Linux发展
 - 2010年，Linux基金会中国区成立
 - 2010年，基金会宣布：在嵌入式领域，Linux排名第一，在服务器、超级计算机、桌面PC持续增长
 - 2018年，Linux基金会宣布成立LF深度学习基金会

Linux内核驱动开发生态

- GNU/Linux
 - GNU: GNU's Not UNIX
 - 自由软件类UNIX操作系统、以GPL方式发布
 - 1983年由Richard Stallman发起，自由软件基金会
 - GNU包括：GNU/Linux、GCC、EMACS、GNU软件包

Linux内核驱动开发方法

- 不同的职位， 不同的技能树
 - 内核开发者
 - 驱动开发者
 - 云、容器、服务器
 - 方案、设备厂商

掌握科学的学习方法

掌握科学的学习方法

- 正确的思想认识

- 现实主义：承认个人局限、抛弃理想主义、速成法
- 实用主义：工作需求、业务驱动
 - 不同的职位，不同的技能树
 - 内核、驱动、云、服务器、方案厂商、设备厂商
- 思维模型：
 - 类比思维：参考其他平台、模仿
 - 经验思维：实验手册、消化吸收
 - 演绎思维：系统化学习、知识体系

掌握科学的学习方法

- 软件工程

- 软件分层思想：

- BSP

- 驱动

- 应用层：驱动、网络服务层、文件系统、虚拟化

- 基础服务：进程、内存管理、VFS、中断、任务调度

- 模块化设计

- 更细粒度的模块划分、模块依赖

- 功能、模块、子系统、框架

掌握科学的学习方法

- 面向对象编程思想
 - OOP是一种编程思想，跟具体的语言无关
 - Linux内核中的OOP思想
 - 结构体
 - 函数指针
 - 设计模式

掌握科学的学习方法

- 本课程规划
 - 基础课程：学习Linux内核基础、API编程
 - 进阶课程一：虚拟化、云、互联网方向
 - 进阶课程二：驱动工程师方向
- 适合哪些人学习：
 - 在校学生、研究生
 - 嵌入式工程师、驱动工程师
 - 对内核感兴趣，有意向从事内核驱动开发
 - Linux运维、系统优化
 - 内核工程师

需要的理论知识和技能

需要的理论知识和技能

- 传统的内核驱动学习路线

- 类比思维:
- 经验思维:
 - 开发板、手册、查字典
 - 显性知识，无法形成知识体系
- 项目驱动



需要的理论知识和技能

- 系统软件编程能力

- 基础的重要性：勿在浮沙筑高台

- 大量的缄默知识、 Know-how、 Know-why
 - 理论基础决定上层技术，认知无法文档化，需要修炼
 - 职位越高的人，也能体会到基础所能决定的上限

- 要有自己的知识体系、思维模型

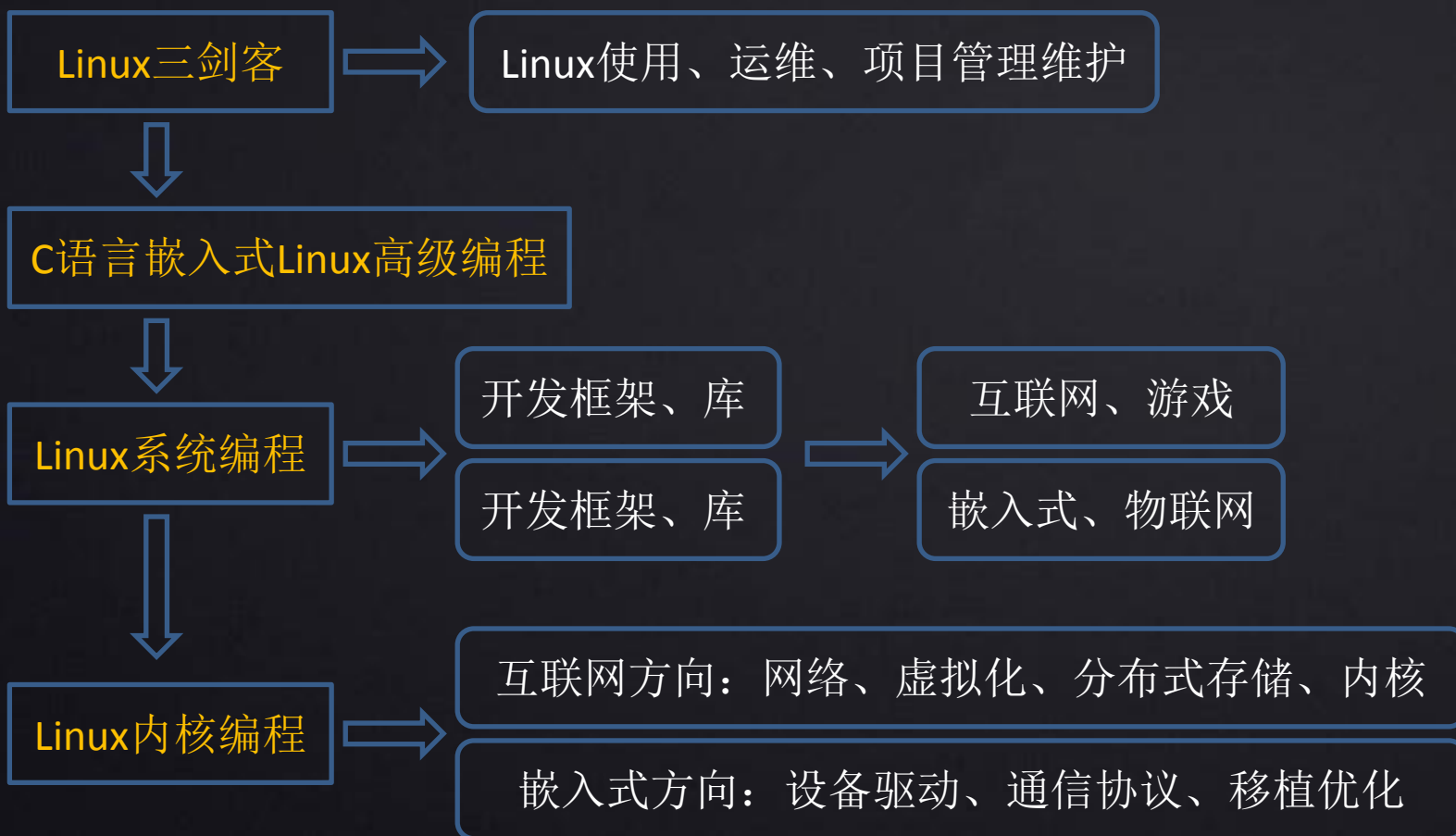
- 高速信息时代，知识是学不完的
 - 跨平台、不断迭代的知识框架
 - 适配不同的平台、开发板、职位、公司...
 - 突破“35岁”中年危机

系统软件开发知识体系

- Linux开发工具、vim、Makefile、IDE、debug
- 工具链：编译、链接、重定位、安装、运行
- 程序主战场：内存管理、堆、栈、堆栈溢出
- 处理器架构、指令集、CPU工作原理
- C语言：结构体、指针、OOP编程思想、扩展语法
- 软件工程：模块化编程、软件分层、框架
- 操作系统概念：进程、线程、调度、临界区、同步
- Linux系统调用：了解、会用、掌握、升华

系统软件开发知识体系

- 学习路线



搭建Linux内核学习平台

搭建学习实验平台

- 小宅实验室简介

- 嵌入式80%的技能都可以脱离开发板学习
- Vmware + Ubuntu + QEMU学习环境
- 不需任何配置，下载后即可运行使用
- 优点
 - 省去硬件的各种干扰，专注软件的业务逻辑
 - 研究U-boot、内核的理想平台
 - 可以更换最新的内核、使用方便
 - 本课程拟使用内核：Linux-5.10

搭建学习实验平台

- 使用说明

- ARM A9 vexpress开发板启动方式
- 启动 U-boot + Linux+ NFS 学习环境
- 手动编译U-boot源码
- 手动编译kernel源码
- 内核镜像下载
 - 关注公众号：宅学部落
 - 输入：小宅实验室，即可获取镜像的网盘下载地址
 - 网盘地址失效，欢迎及时反馈，获取最新下载地址


内核版本的选择

内核版本的选择

- 内核的不同版本

- **mainline**: 由Linus维护, 每2~3月发布一次新版本
- **RC1~RC5**: 待发布(pre-release)内核, 添加了新的功能, 在进入stable版本前需要测试、bug fix
- **stable**: 每周发布一次, bugfix backporting

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Release
5.9.6 

mainline:	5.10-rc2	2020-11-01	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]	
stable:	5.9.6	2020-11-05	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
stable:	5.8.18 [EOL]	2020-11-01	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	5.4.75	2020-11-05	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.19.155	2020-11-05	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.14.204	2020-11-05	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.9.241	2020-10-29	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	4.4.241	2020-10-29	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
linux-next:	next-20201106	2020-11-06						[browse]

内核版本的选择

- 内核的不同版本

- Linux-next: CI 提前bugfix测试, 由Stephen Rothwell 维护
- LTS: Long Time Service, 5年的维护周期

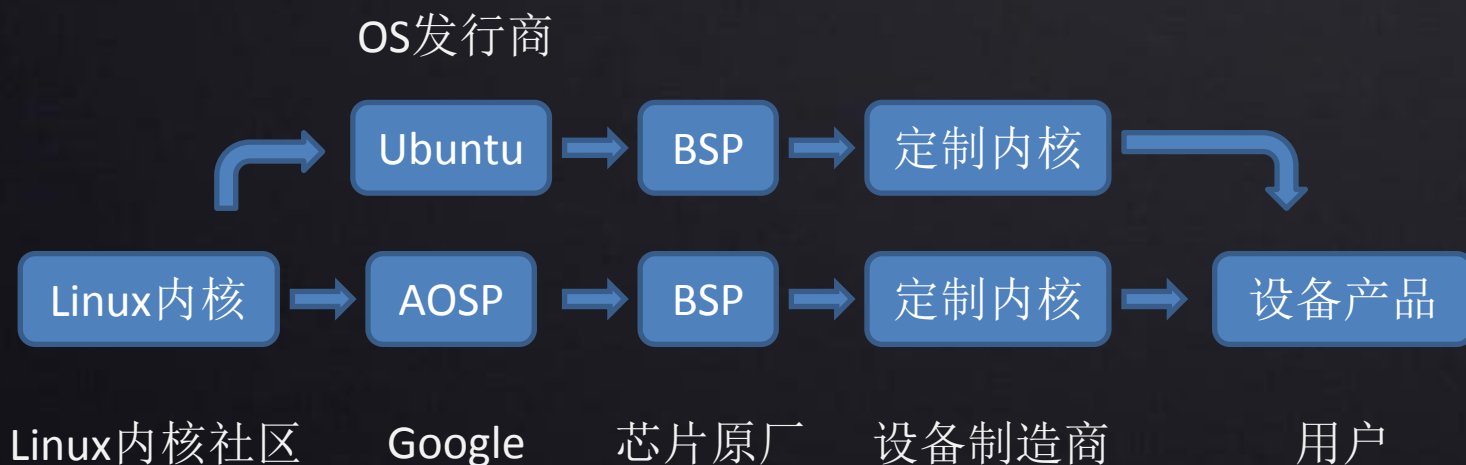
Version	Maintainer	Released	Projected EOL
5.4	Greg Kroah-Hartman & Sasha Levin	2019-11-24	Dec, 2025
4.19	Greg Kroah-Hartman & Sasha Levin	2018-10-22	Dec, 2024
4.14	Greg Kroah-Hartman & Sasha Levin	2017-11-12	Jan, 2024
4.9	Greg Kroah-Hartman & Sasha Levin	2016-12-11	Jan, 2023
4.4	Greg Kroah-Hartman & Sasha Levin	2016-01-10	Feb, 2022

内核版本的选择

- 下游厂商的内核版本
 - SLTS: Super Long Time Service, 10~20年的维护周期
 - OS发行商各自维护的内核版本
 - Android kernel
- 新版本内核特性
 - 虚拟化、云
 - 容器技术、namespace、cgroup
 - 用户态I/O、DPDK、eBPF
 - 新的内核调试工具、打印工具

内核版本的选择

- 内核版本选择
 - 互联网、云
 - OS发行商
 - Android kernel
 - 芯片原厂、设备制造商

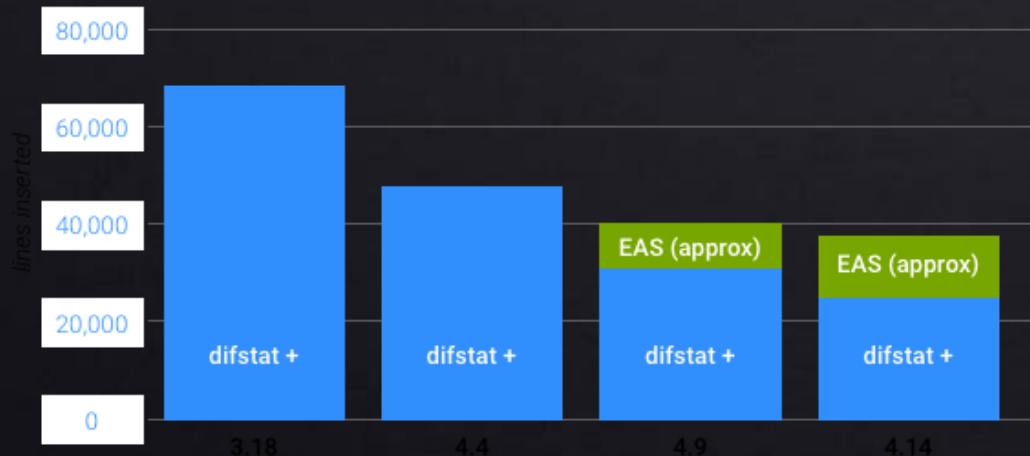


Android内核

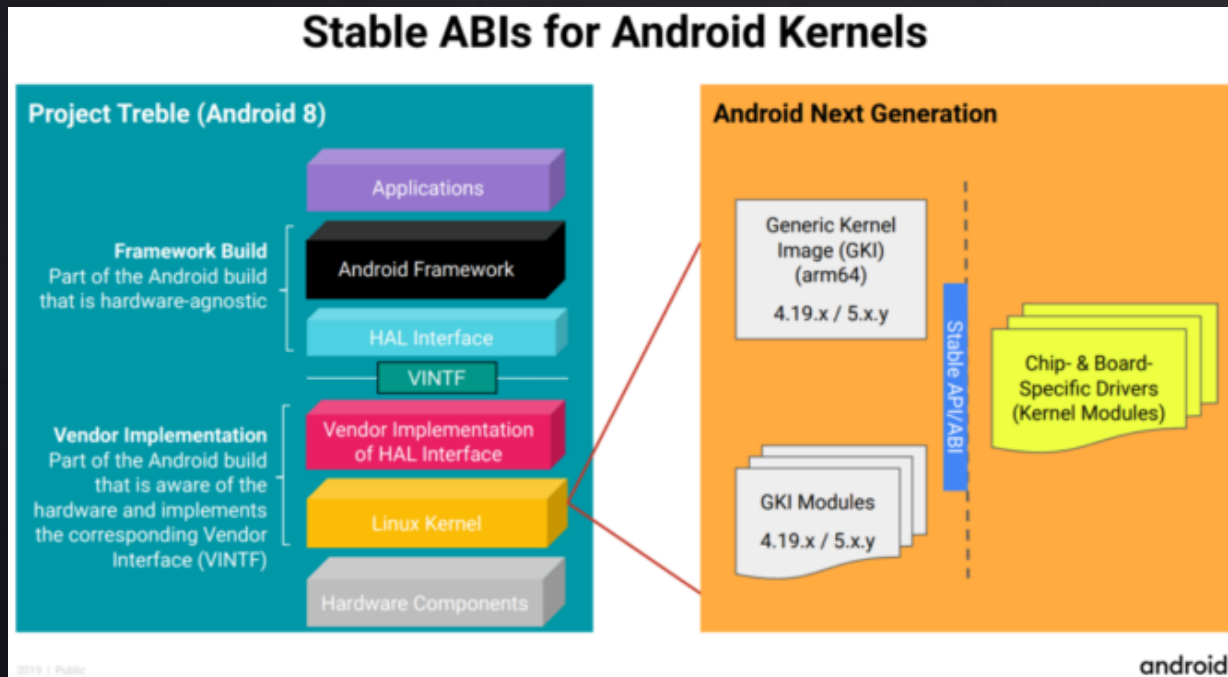
- Android-4.4.y对linux-4.4的修改
 - 新增32266行，删除1546行，替换255行
 - 19.8% Energy Aware Scheduling (kernel/sched)
 - 13.8% 网络 (net/netfilter)
 - 13.5% Sdcardfs (fs/sdcardfs)
 - 9.4% USB (drivers/usb)
 - 7.2% SoC (arch/arm64, arch/x86)
 - 6.1% 输入 (drivers/input/misc)
 - 5.4% FIQ 调试 (drivers/staging/android/fiq_debugger)
 - 3.6% Goldfish 模拟器 (drivers/platform/goldfish)
 - 3.4% Verity (drivers/md)
 - 11.6% 其他

内核版本的选择

- Android通用内核
 - 内核碎片化问题：不同版本、不同设备厂商
 - 内核模块化：
 - 系统芯片、设备外设、板属组件
 - 通用内核直接升级，android-X.Y与 LTS patch定期合并
 - 内核加固
 - 与上游内核的差异逐渐缩小



Android通用内核



交叉编译器的选择：GCC

交叉编译器的选择

- GCC

- GNU C Compiler → GNU Compiler Collection
- 包括：编译器、链接器、binutils、glibc、头文件
- 1987年发布第一个版本
- 支持多种语言：C、C++、Java、Go等
- 支持多种硬件平台：X86、ARM、MIPS、RISC-V等

交叉编译器的选择

- 交叉编译器
 - GNU C Compiler for ARM Architecture
 - 安装: `# apt install gcc-arm-linux-gnueabi`
 - 安装: `# apt install gcc-arm-linux-gnueabihf`
 - `gcc-arm-linux-gnueabi`
 - GNU C compiler for armel architecture
 - `gcc -mfloat-abi softfp`
 - `gcc-arm-linux-gnueabihf`
 - GNU C compiler for armhf architecture
 - `gcc -mfloat-abi hard`

交叉编译器版本

- 浮点运算
 - FPU: floating point unit
 - 浮点寄存器
- 编译器的三个版本:
 - soft:
 - 不使用FPU进行浮点运算, 适合早期无FPU的处理器
 - softfp
 - 使用FPU, 但使用普通寄存器传参
 - 参数需要转换成浮点再计算, 中断负荷小
 - 编译器使用该版本, 兼容没有FPU的处理器
 - hard:
 - 使用PFU, 使用浮点寄存器传参, 中断负荷大, 性能高

交叉编译器版本

- 不同命名的编译器
 - arm-linux-gcc: 针对 arm+MMU+Linux
 - arm-elf-gcc: 针对 arm+uclinux
 - arm-none-gnueabi-gcc
 - arm-none-linux-gnueabi-gcc
- 编译器厂商
 - ARM
 - IAR: 瑞典的一家公司
 - GNU
 - Linaro
 - CodeSourcery

交叉编译器的选择: Clang

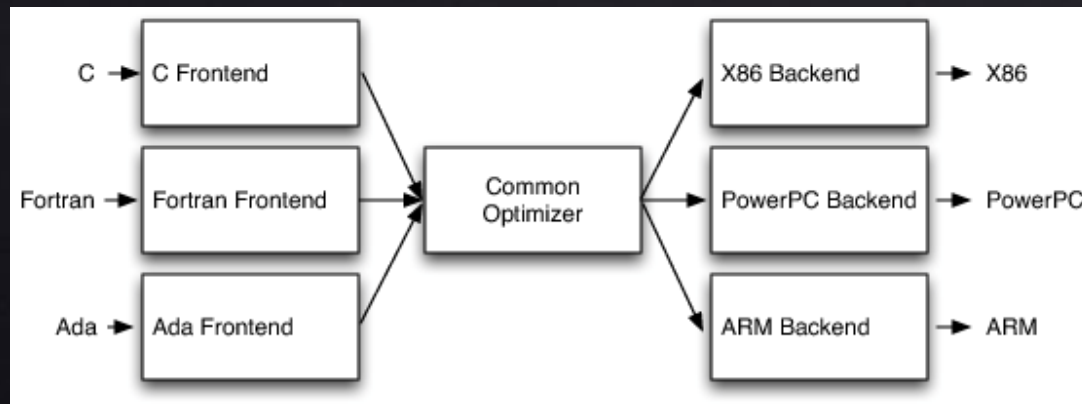
编译器的发展史

- 编译器的构成

- Frontend: 源码分析、语法检查, 输出中间代码
- Optimizer: 对中间代码进行优化、使其运行更高效
- Backend: 将中间代码转换为某一个平台的机器代码

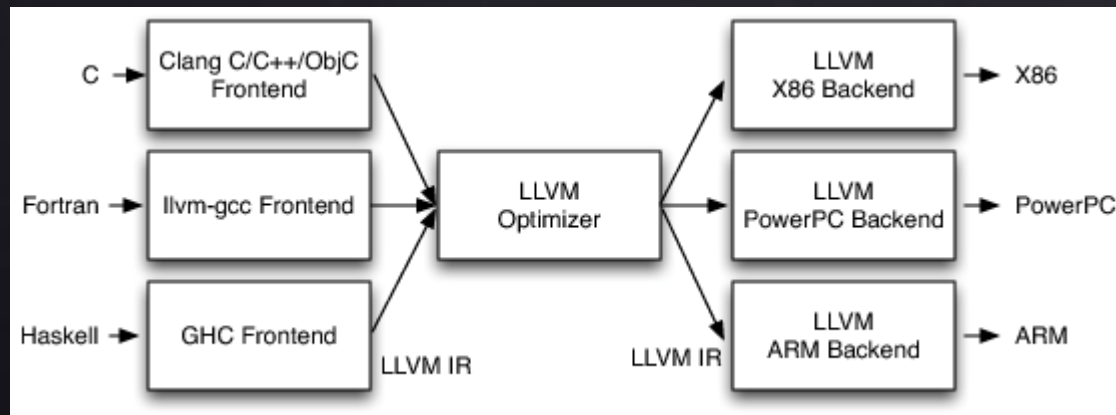
- 编译器的实现

- GCC: 前端和后端分离, 支持多种语言、多个平台
- 缺点: 耦合高, 代码可重用性低



交叉编译器的选择

- LLVM
 - Low Level Virtual Machine
 - 一个开源的编译器框架
 - 模块化设计、代码可重用性高
 - 中间语言LLVM IR，类C表达，可扩展各种前端、后端
 - 支持C/C++/Java/Fortran/Ada/Swift/Python/Ruby/Rust/C#



交叉编译器的选择

- LLVM编译器实例
 - LLVM GCC：前端使用GCC
 - Apple的Obj-C特性需要GCC即时更新和支持
 - GCC代码耦合性高，不利于IDE集成、模块式调用
 - Apple公司的编译器：GCC → LLVM GCC → Clang
 - Clang：前端使用Clang
 - 基于LLVM开发的C/C++/Obj-C编译器
 - 目前广泛应用于苹果软件生态、Android、Fuchsia...
 - 方舟编译器
 - 基于LLVM，用于构建鸿蒙软件生态
 - 编译优化

Linux内核编码风格

Linux内核编码风格

- 不同的编码风格
 - Windows: ReadData()
 - Linux: read_data()
 - GNU: read_data ()
 - Linux内核: Documentation/process/coding-style.rst
 - 公司
 - 社区
 - 项目

Linux内核编码风格

- 代码缩进与空格

```
static __always_inline
void arch_timer_reg_write_cp15(int access, enum arch_timer_reg reg, u32 val)
{
    if (access == ARCH_TIMER_PHYS_ACCESS) {
        switch (reg) {
            case ARCH_TIMER_REG_CTRL:
                asm volatile("mcr p15, 0, %0, c14, c2, 1" : : "r" (val));
                break;
            case ARCH_TIMER_REG_TVAL:
                asm volatile("mcr p15, 0, %0, c14, c2, 0" : : "r" (val));
                break;
        }
    } else if (access == ARCH_TIMER_VIRT_ACCESS) {
        switch (reg) {
            case ARCH_TIMER_REG_CTRL:
                asm volatile("mcr p15, 0, %0, c14, c3, 1" : : "r" (val));
                break;
            case ARCH_TIMER_REG_TVAL:
                asm volatile("mcr p15, 0, %0, c14, c3, 0" : : "r" (val));
                break;
        }
    }
    isb();
}
```

Linux内核编码风格

- 代码缩进与风格

- 使用tab键缩进，一般是8个空格
- 在if、switch、case、for、do、while后加空格，大括号不能单独占一行
- 二、三元运算符两侧一般要加一个空格：+ - / |
- sizeof、typeof、alignof、__attribute__后不要加空格
- 小括号里表达式的两侧不要加空格
- 一元运算符、函数名后面一般不加空格
- 函数的大括号单独占一行
- 函数参数一行写不下，可以换行写

Linux内核编码风格

- 函数、变量命名
 - 不用大写: `copy_to_user`、`copy_from_user`
 - 少用`typedef`, 不直观的数据类型, 降低代码可读性
 - 宏定义使用大写格式, 形如函数的宏可以用小写
 - 函数尽量短、半屏或一屏显示完、各个函数用空行隔开
 - 统一的函数退出路径: `goto`, 错误统一处理

Linux内核编码风格

- 格式化工具:indent
 - 安装: `# apt install indent`
 - 使用: `# indent -ce hello.c` else放在}后面
 - 默认GNU风格: `-gnu`
 - K & R风格: `-kr`
 - Linux风格: `-linux`
 - Berkeley风格: `-orig`
 - 配置文件: `.indent.pro`
 - Linux内核代码格式化: `scripts/Lindent`

Linux内核编码风格

- 格式化工具:indent

indent参数	值	说明
--blank-lines-after-declarations	bad	变量声明后加空行
--blank-lines-after-procedures	bap	函数结束后加空行
--braces-after-if-line	bl	"if"和"{"分做两行
--brace-indent 0	bli0	"{"不继续缩进
--case-indentation 0	cli0	switch中的case语句所进0个空格
--cuddle-else	nce	"else"和其前面的"}"另起一行
--space-after-procedure-calls	pcs	函数和"("之间插入一个空格
--space-after-for	saf	for后面有空格
--space-after-if	sai	if后面有空格
--tab-size	ts4	一个tab为4个空格

内核模块的编译与运行

内核模块化编译与运行

- 宏内核与微内核
 - 宏内核：
 - 微内核：调度，驱动运行在用户态
- 最简单的内核模块
 - hello.c
 - Makefile

内核调试

打印内核信息

- 打印等级
 - 控制台打印等级
 - printk打印等级
- 不同级别的打印函数
 - <linux/printk.h>
 - pr_err、pr_warn
 - pr_notice、pr_info
 - 驱动打印
 - dev_err、dev_warn
 - dev_info、dev_debug

打印内核信息

- 跟踪异常
 - `dump_stack()`: 打印调用栈
 - `WARN(condition, format...)`: 类似 `printk`
 - `WARN_ON(condition)`: 调用 `dump_stack`, 不会OOPS
 - `BUG()`: 内核界的断言, 触发内核OOPS, 输出log
 - `BUG_ON(condition)`: 条件触发内核OOPS, 输出log
 - `panic(fmt...)`: 系统crashed, 输出log
 - 底层实现: 跟不同CPU的体系结构相关

Linux内核之旅

纸上得来终觉浅
绝知此事要躬行



TIPS

- 更多Linux内核学习资源

- 免费在线视频: <https://space.bilibili.com/382223675>
- 免费在线文本教程: www.zhaixue.cc
- 进阶视频: <https://wanglitao.taobao.com/>
- 文档代码仓库: https://gitee.com/wang_litao
- QQ群: 475504428
- Qemu虚拟开发板镜像及本视频下载:
 - 关注公众号, 输入: 小宅实验室



Linux内核编程 预售地址: [Linux内核编程 嵌入式工程师自我修养系列教程第4步](#)