RUST语言与RISC-V操作系统

洛佳

华中科技大学 网络空间安全学院

2020年8月29日

演讲内容

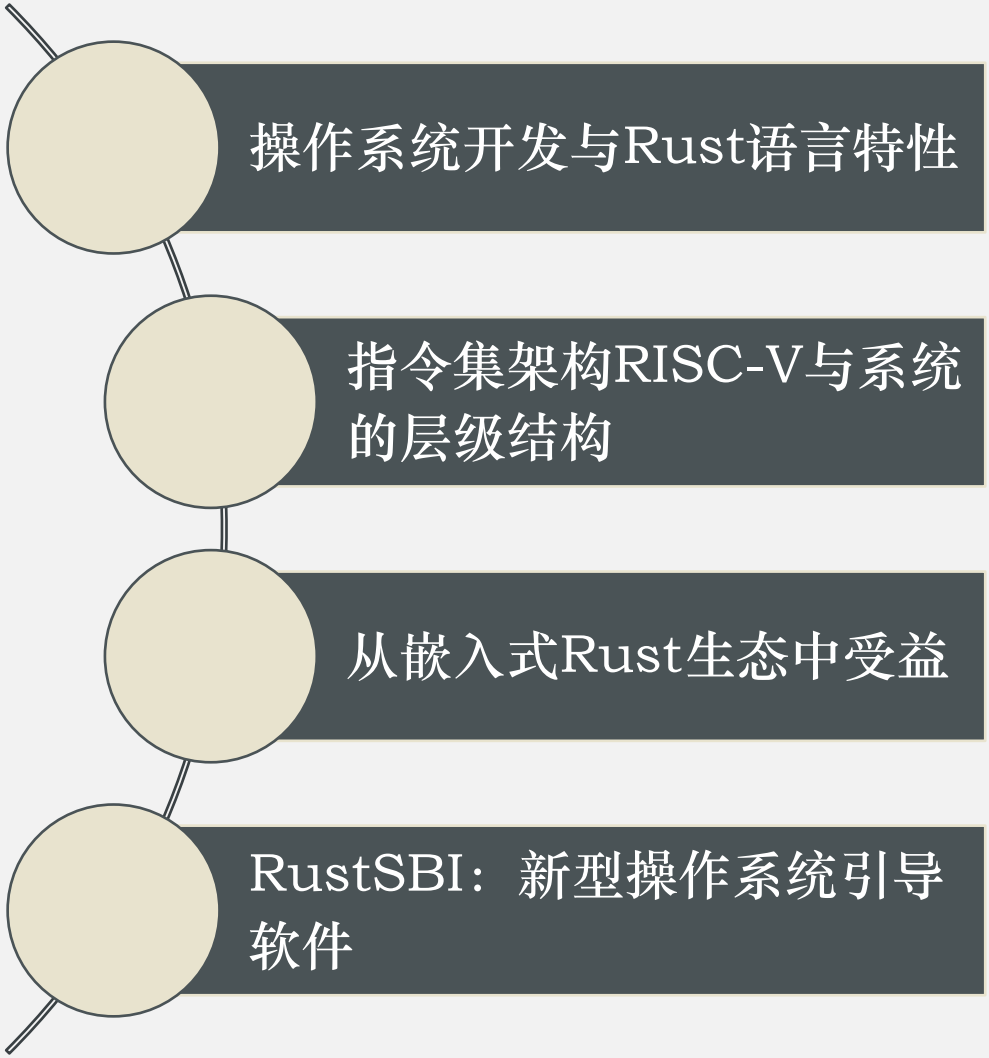
开源软件点亮计划 鹏城实验室

“下一代Rust操作系统：zCore（RISC-V）”

暑期活动总结报告

洛佳

华中科技大学 网络空间安全学院



关于我

爱生活，爱Rust

笔名是洛佳

- 姓名蒋周奇，科普作者。翻译《编写Rust语言的操作系统》
- 生于1999年6月，热爱民族乐器

长期贡献开源社区

- 学习Rust三年余，热爱Rust嵌入式与操作系统生态
- 开发“GD32V”系列和“RV32M1”系列嵌入式处理器支持库

有商业项目经历

- 参与开发“核能”和“科洛桑”游戏引擎
- 曾与网易游戏商业合作

编写RUST语言的操作系统

体验一回21世纪的工程设计！

RUST语言：属于21世纪的语言新星

性能好

- 极小运行时，无垃圾回收设计
- 可适用于各类嵌入式设备

可靠性强

- 类型系统丰富，所有权模型
- 移动语义，内存、线程安全设计良好

生产效率高

- 文档齐全，编译器提示友好、有帮助
- 提供通用的包管理器

例子：接口与抽象

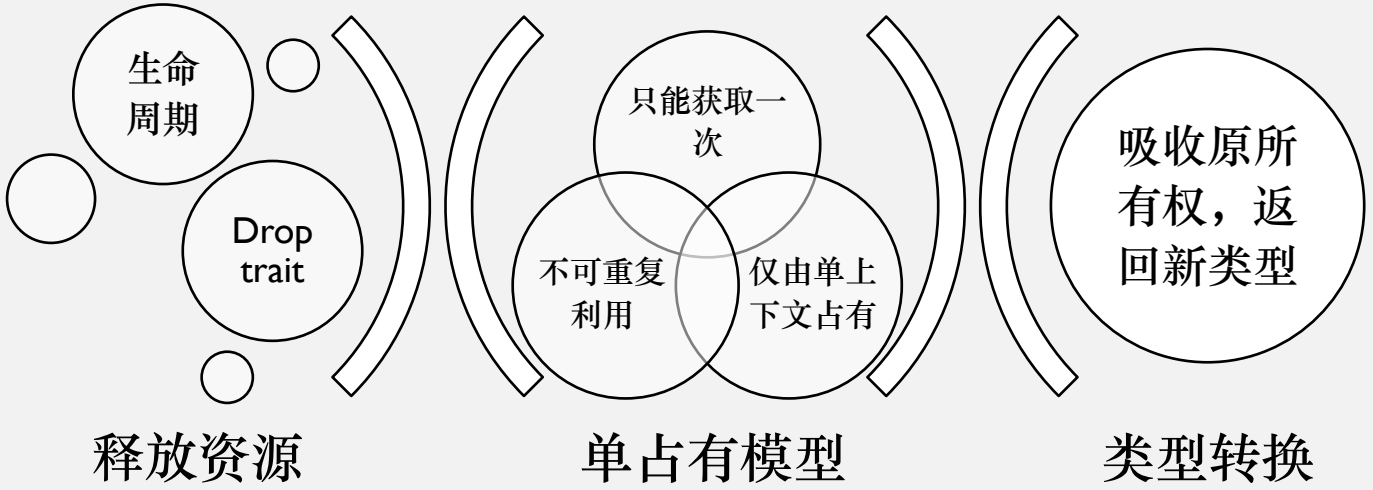
传统的编程语言

- 使用虚函数表或特别编写结构体
- 传递函数指针
- 用户自己编写链表

全新的Rust语言

- 使用独特的“trait”作为抽象方式
- 零成本使用闭包语法
- 标准库“alloc”包

RUST的语法适合开发操作系统



强大的RUST宏：库的组成部分

卫生宏、过程宏

- 内部展开，不影响上下文
- “包裹”代码
- 自定义语法

与Rust语法结合

- 给定对齐；指定代码区
- 导出给其它库使用

两门语言的宏语法对比

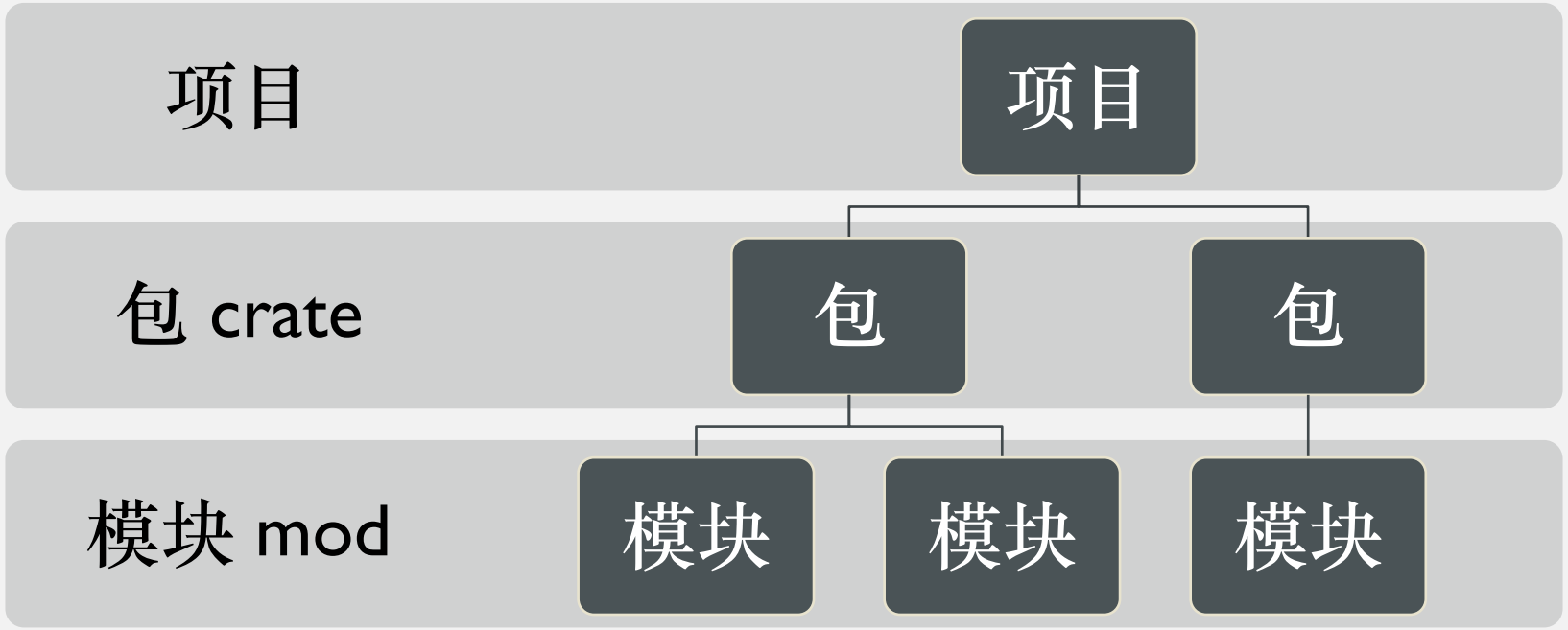
RUST语言

```
use library_name::dll_main;
macro_rules! vec { ... };
// 解析语法树（1..5部分），返回变量值
let v = vec![1, 2, 3, 4, 5];
println!("第一个数字是: {}", v[0]);
// 输入语法树（整个函数），输出语法树
#[dll_main] fn dll_entry() { ... }
```

C语言

```
#include "library_name.h"
#define PB(val) v.push_back()
// 文本替换较长的语句
vector<int> v; PB(1); ...; PB(5);
printf("First num: %d\n", v[0]);
// 文本替换为编译器定义的额外注解
DLL_MAIN void dll_entry() { ... }
```

模块化编程



丰富的工具链和生态

模块化编程的语法支持

- 关键字mod和crate；可见性pub和pub(crate)

实用而统一的工具链

- Rust编译目标丰富，支持嵌入式编译目标
- 统一的包管理工具Cargo

RISC-V指令集和层级操作系统结构

如果你有解决不了的问题，不妨多加一层

RISC-V指令集架构简介

精简指令集

- 最基础的扩展只有40多条指令
- 寻址方法简单
- 单独的浮点指令集
- 很适合新手学习

CSR寄存器

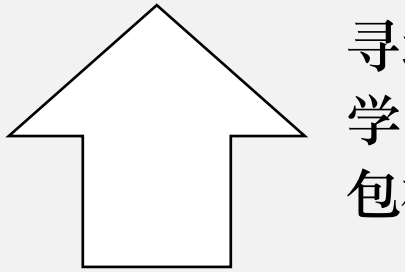
- 开启和关闭中断
- 陷入处理，包括异常和中断
- 访问不同特权级内存
- 专门的读写指令

特权级操作

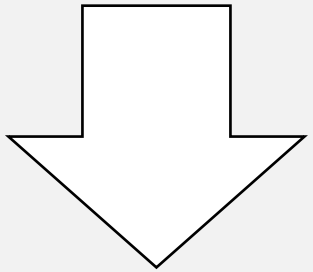
- 虚拟内存和页表刷新指令
- 指令缓存刷新指令
- 进入其它特权级
- 调用运行环境

RISC-V与成熟架构的简单比较

与X86比较

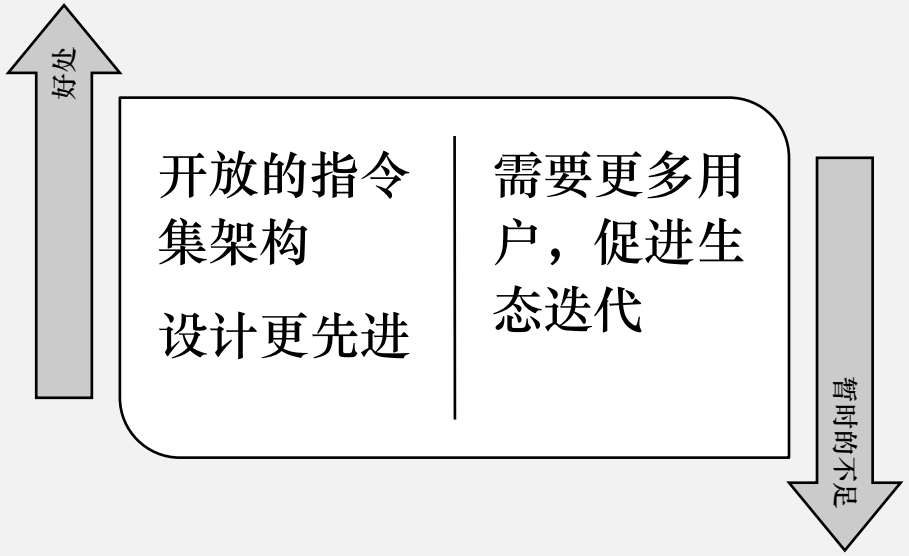


寻址方式简单，学习容易，历史包袱少

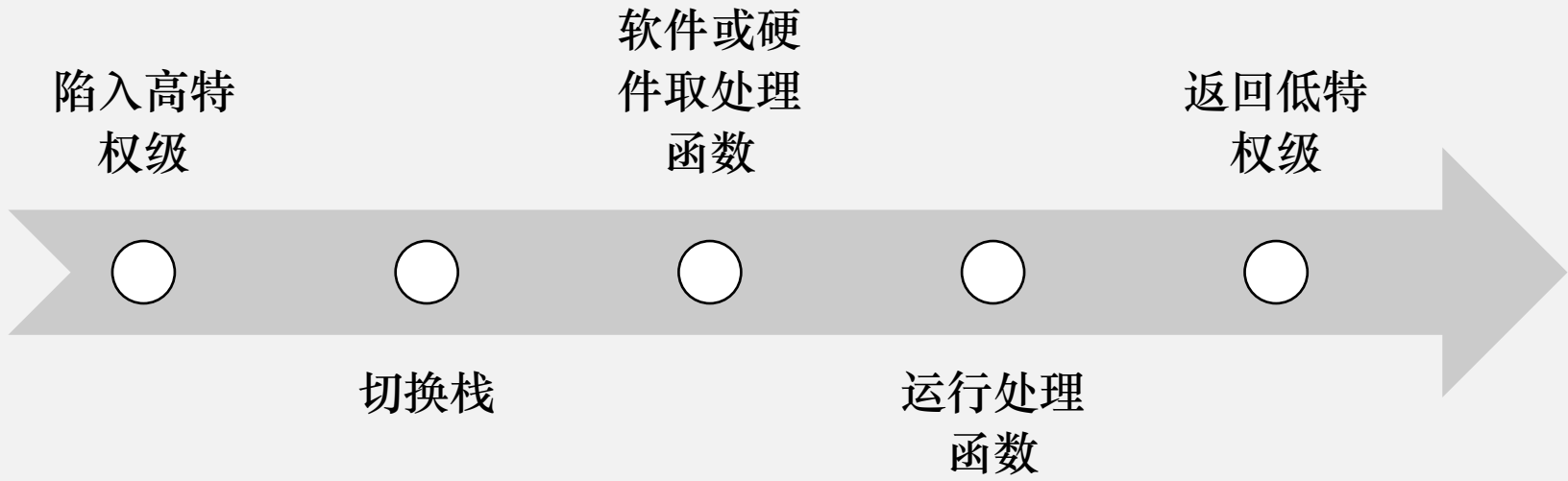


需要更多的厂商和学习者投入时间

如果是ARM指令集呢



切换上下文



虚拟内存和内存保护

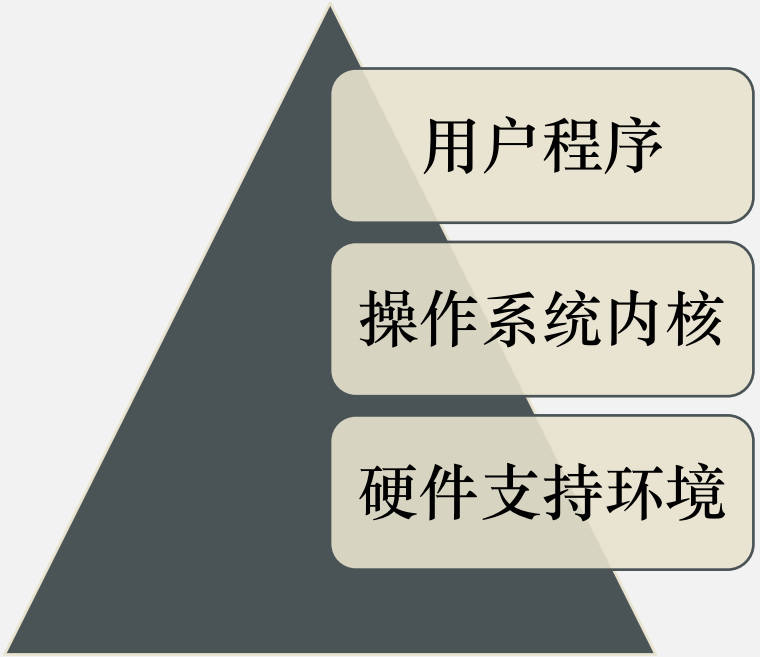
pmp寄存器

- 一组共16个寄存器
- 机器特权级简单内存保护
- 简单的嵌入式处理器、复杂的桌面处理器都会配备

satp寄存器

- 地址编号，顶级物理页号
- 系统特权级的复杂机制
- 虚拟内存和内存分页
- Sv32、Sv39等页表标准
- 较为复杂的桌面处理器

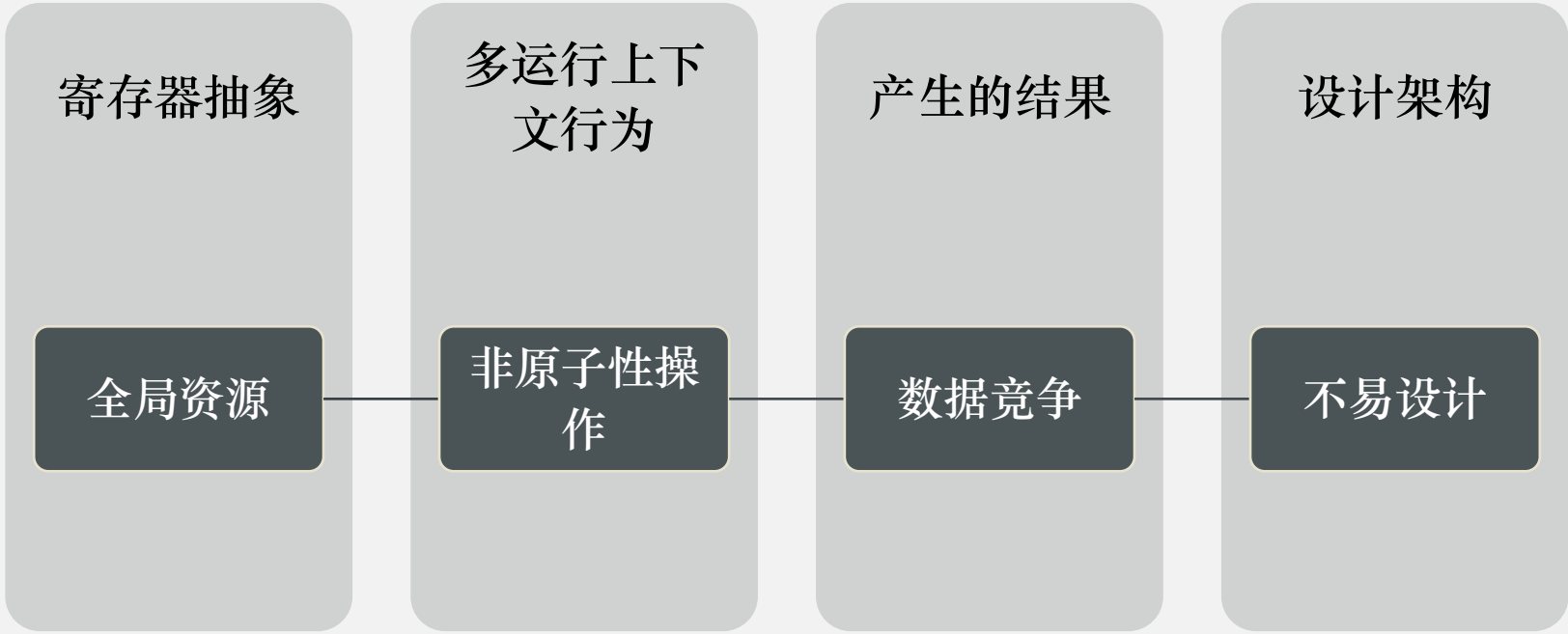
我们的操作系统分三层



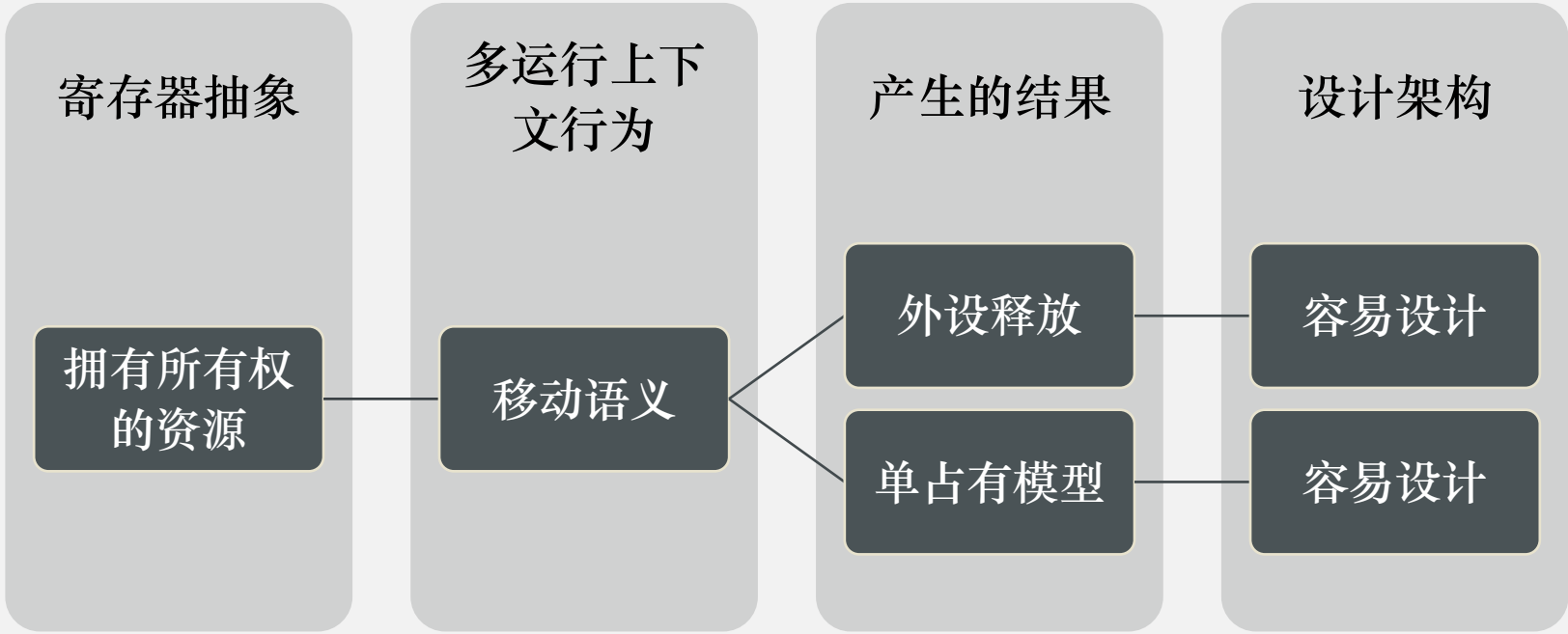
从嵌入式RUST生态中受益

这就是我为什么热爱Rust

例子：外设抽象——旧的C语言模型

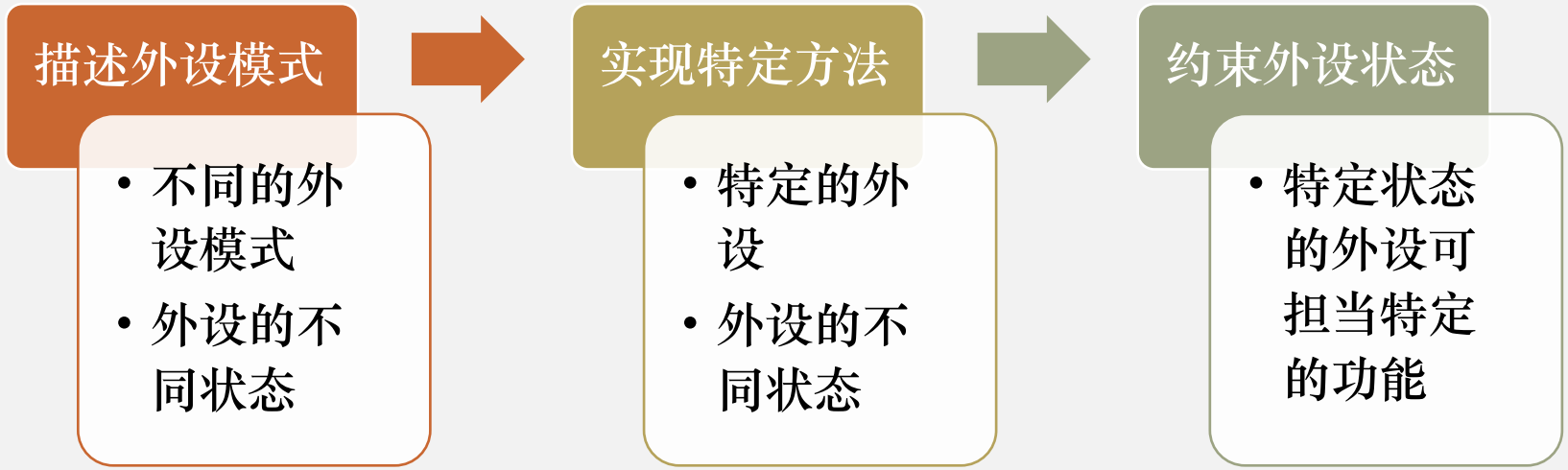


例子：外设抽象——新的RUST设计方式



*此外也有特殊的类型系统设计

用类型系统描述硬件状态



EMBEDDED-HAL：统一的外设抽象库

标准统一

使用Rust语言

针对外设本身特性的抽象

实现由实现库完成

生态圈庞大

支持海量市售芯片，如K210

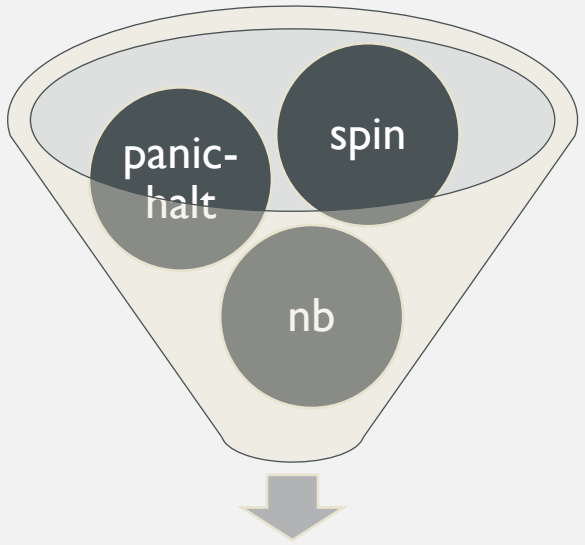
对片内外设、外挂外设都能支持

编码容易

模块间衔接嵌套便捷，能整合相关项目

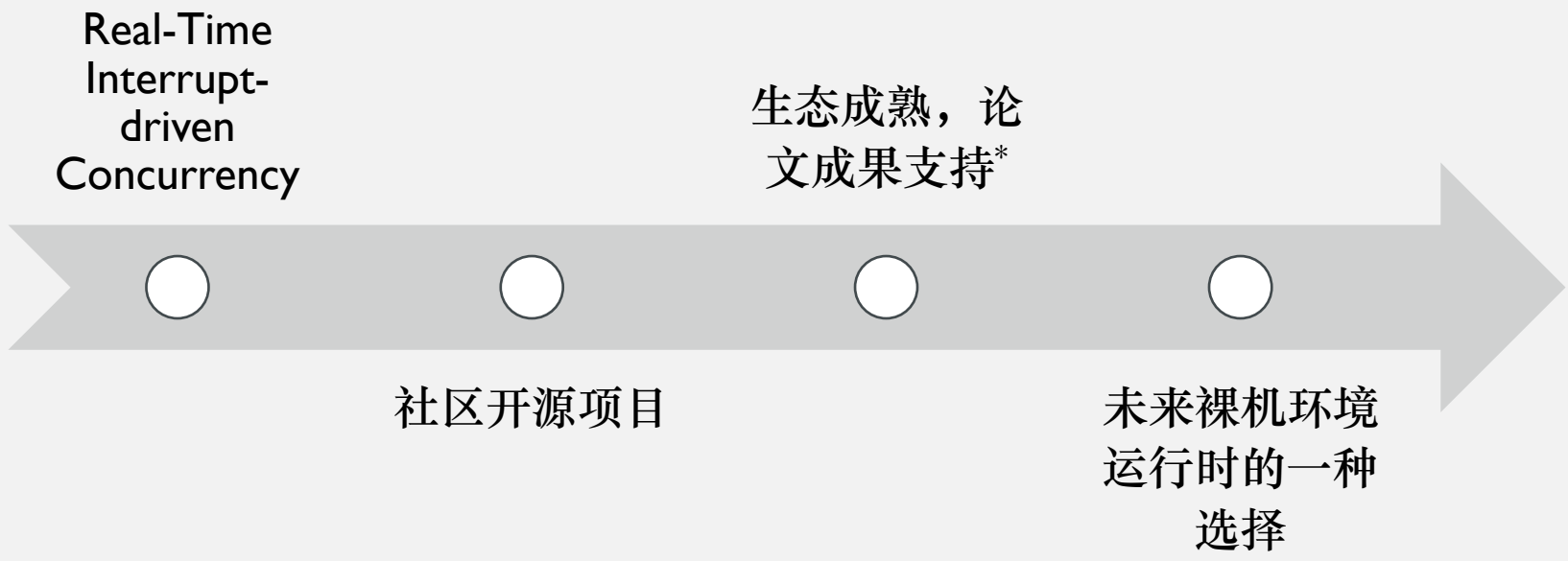
很容易为新芯片编写支持库

RUST语言丰富的裸机生态



提供操作系统使用

RTIC：中断支持的裸机并发运行时



* Eriksson, J., et.al. (2013, June). Real-time for the masses, step 1: Programming API and static priority SRP kernel primitives. In Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on (pp. 110-113). IEEE.

为什么嵌入式RUST开发中少有BSP的概念

RUSTSBI：新型操作系统引导软件

OpenSBI哪都好，就一个缺点：它是用C语言写的

什么是SBI?

引导程序

- 启动系统内核
- 收集设备信息，提供给操作系统
- 类似于UEFI

统一的硬件环境

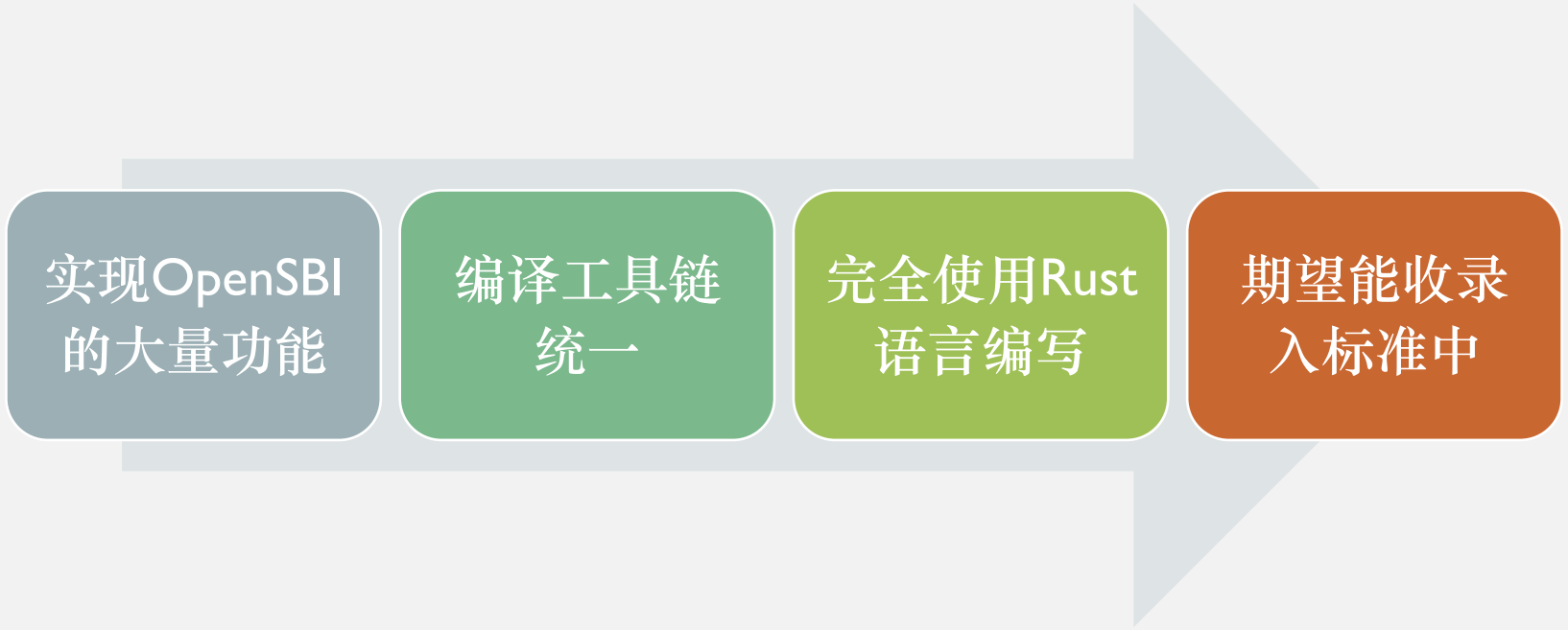
- 通过系统调用，提供实用功能
- 监控所有的处理器核
- 发送跨核软中断
- 提供兼容性支持

是一个标准

- 适用于RISC-V
- 期望消除部分硬件差异
- 它有多个实现，如OpenSBI

SBI标准的设计思路

欢迎使用RUSTSBI



为什么选用RUST语言开发SBI实现

嵌入式Rust生态圈

相比在C语言项目外面包装一层

容易支持新硬件

编译期语义约束强

SBI能实现的兼容性设计

K210芯片与SBI的兼容性设计

快表刷新指令

- 1.11版的sfence.vma与1.9版的sfence.vm
- M层的SBI实现捕获指令异常
- 用旧指令模拟新指令
- 同样的方法也用于rdtime指令，不存在的CSR情况

修改的CSR位

- 1.9版的mstatus.VM
- 1.11版在satp里面
- 每次刷新页表时，启用这些位
- 有一定局限性，未来的芯片可尝试其它设计，比如把CSR的修改作为某个外设中断等等

总结与未来展望

Rust语言与RISC-V操作系统

致谢

- 感谢向勇教授和陈渝教授，提供这次实习活动的机会。感谢鹏城实验室和李睿老师提供指导。
- 感谢王润基学长提供的灵感，这为本次项目提供很大的帮助。感谢吴一凡学长和我交流。
- 感谢我的小组成员，他们是：石伟、徐文浩、车春池、周鹤洋同学。小组成员的帮助为我提供了不同的方法和思路。
- 感谢社区的张汉东老师，张老师是《Rust编程之道》的作者，也为我的社区工作提供了非常多的帮助。

谢谢各位

Rust语言与RISC-V操作系统

洛佳

华中科技大学 网络空间安全学院