

****

课题名称：支持自动加载的跨操作系统的USB驱动

学生姓名：姚宏伟\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2024年6月

目 录

[1. 课题背景与意义 3](#_Toc17062)

[1.1 课题背景 3](#_Toc10391)

[1.2 课题意义 3](#_Toc28198)

[2. 国外或国内研究现状 3](#_Toc14209)

[2.1 国外研究现状 3](#_Toc19653)

[2.2 国内研究现状 4](#_Toc15021)

[2.3 现有研究的不足之处 4](#_Toc8912)

[3. 课题预期成果 4](#_Toc25609)

[3.1 项目目标 4](#_Toc17896)

[3.2 预期成果 4](#_Toc14427)

[3.3 项目进度安排 4](#_Toc16062)

[4. 需求分析 5](#_Toc24718)

[4.1 用户需求分析 5](#_Toc12714)

[4.2 系统需求分析 5](#_Toc23137)

[5. 系统设计 5](#_Toc3338)

[5.1 关键技术 5](#_Toc11222)

[5.2 功能设计 5](#_Toc13195)

[6. 参考文献 5](#_Toc5840)

# 课题背景与意义

## 1.1 课题背景

随着计算机技术和硬件的发展，USB（Universal Serial Bus）已成为计算机和其他电子设备之间通信和连接的标准接口。USB驱动程序在这些设备之间的通信中起着至关重要的作用。然而，随着国产化的兴起，无数国产操作系统开始涌现，USB作为一种较为复杂的接口(甚至可以确切的说，是最复杂的总线)，其驱动开发起来较为困难，而且传统的USB驱动程序通常针对特定的操作系统进行开发，这导致了跨操作系统的兼容性问题。

当前的USB驱动开发主要面临以下几个挑战：

* **操作系统依赖性强：**大多数现有的USB驱动程序是为特定的操作系统设计的，难以在不同操作系统之间移植。
* **维护和更新复杂：**由于驱动程序的操作系统依赖性，每次操作系统更新都可能导致驱动程序的兼容性问题，增加了维护和更新的复杂性。
* **安全性和稳定性问题：**不同操作系统的底层架构和驱动模型不同，导致驱动程序在跨操作系统使用时可能出现安全性和稳定性问题。

针对这些问题，本课题提出了一个跨架构、跨操作系统的USB HOST驱动框架。该框架支持当前主流的HOST控制器版本（如XHCI），并通过使用Rust编程语言实现，以充分利用Rust的安全性和可移植性。

## 1.2 课题意义

本课题的研究具有重要的理论价值和实际应用意义：

* **提高设备兼容性：**开发一个跨操作系统的USB驱动框架，可以大幅提高不同设备之间的兼容性，使得相同的驱动程序能够在多个操作系统上运行，减少了开发人员的工作量。
* **简化驱动开发流程：**通过构建一个操作系统无关的抽象层，开发人员可以更加专注于驱动程序的功能实现，而无需关心不同操作系统之间的差异，大大简化了驱动开发流程。
* **提升系统安全性和稳定性：**Rust语言具有内存安全和并发安全的特点，使用Rust开发驱动程序可以有效避免常见的内存泄漏和竞争条件等问题，提升系统的安全性和稳定性。
* **模块化加载能力：**本框架支持驱动模块的按需加载，这不仅提高了系统的灵活性和可扩展性，还能够根据实际需求动态加载所需的驱动模块，优化资源使用。
* **推动行业标准化发展：**跨操作系统的USB驱动框架有助于推动USB驱动开发的标准化进程，为行业提供一个通用的解决方案，具有广泛的应用前景和市场价值。

# 2. 国外或国内研究现状

## 2.1 国外研究现状

在国际上，关于跨操作系统的USB驱动开发，已有一些研究成果和应用案例：

* **USBFuzz框架**：由Peng和Payer等人提出，USBFuzz是一种通过设备仿真进行USB驱动模糊测试的框架。该框架能够有效地检测USB驱动中的漏洞，提升驱动的安全性和稳定性。
* **高性能同步数据传输系统**：Huang和Wang等人研究并应用了基于USB 3.0外设控制器的高速可调同步数据传输系统，这项研究在高效数据传输和处理方面有显著贡献。
* **Rust在嵌入式软件开发中的应用**：Borgsmüller的博士论文中，详细探讨了Rust编程语言在嵌入式系统开发中的应用，强调了Rust在安全性和性能方面的优势。

这些研究展示了国际上在USB驱动开发领域的先进技术和应用，但多数研究仍然集中于特定操作系统或特定硬件平台上。

## 2.2 国内研究现状

在国内，随着国产操作系统和自主可控硬件的发展，相关领域的研究也取得了一定成果：

* **国产操作系统生态体系建设**：韩乃平和李蕾等人分析了国产操作系统生态体系的建设现状，指出了当前国产操作系统在驱动开发方面的不足和挑战 。
* **XHCI协议软核IP设计**：杨志荣、胡茂海和董理等人设计了FPGA的XHCI协议软核IP，并进行了实际应用，展示了XHCI协议在国产硬件中的应用前景 。
* **嵌入式处理器高性能中间件软件设计**：张帅林的研究中，开发了基于嵌入式处理器的高性能中间件软件，展示了在嵌入式系统中提高性能和可移植性的方法 。

这些研究反映了国内在USB驱动和相关技术领域的进展，但在跨操作系统的驱动开发和动态模块加载方面仍存在技术空白。

## 2.3 现有研究的不足之处

尽管国内外在USB驱动开发领域已取得不少成果，但仍存在以下不足：

* **技术局限性：**现有大多数研究和应用都局限于特定的操作系统和硬件平台，缺乏跨平台、跨操作系统的兼容性。
* **应用场景受限：**许多研究集中于特定场景或需求，缺乏通用性和扩展性，难以满足不同操作系统和硬件架构的需求。
* **动态模块加载：**目前尚无成熟的框架能够支持USB驱动的按需加载，现有解决方案多为静态加载，灵活性不足。

本课题提出的跨操作系统的USB HOST驱动框架，旨在解决上述问题，通过使用Rust语言实现操作系统无关的驱动框架，并引入动态模块加载机制，提高驱动程序的兼容性、灵活性和安全性。

# 3. 课题预期成果

## 3.1 项目目标

本课题的总体目标是开发一个支持自动加载的跨操作系统USB HOST驱动框架，框架目前所支持的HOST控制器版本为XHCI。具体目标包括：

* 跨架构支持：设计和实现一个支持多种硬件架构（如x86和ARM）的USB HOST驱动框架，以确保驱动程序在不同硬件平台上的通用性和兼容性。
* 跨操作系统支持：实现驱动框架在多个操作系统（如Windows、Linux、macOS）上的兼容性，确保驱动程序能够在不同操作系统环境中无缝运行。
* 自动加载功能：支持驱动模块的按需动态加载，以提高系统的灵活性和资源利用效率。
* 使用Rust编程语言：利用Rust语言的安全性、性能和可移植性，编写操作系统无关的驱动程序，确保驱动程序在多个操作系统和硬件平台上运行的安全性和稳定性。
* 示范性Demo：提供一个完整的示范性Demo，展示驱动框架的基本功能和跨平台能力，包括基本的设备识别、数据传输和错误处理。

## 3.2 预期成果

通过本课题的研究和开发，预期将取得以下成果：

* 技术成果
  + 跨操作系统USB HOST驱动框架：一个支持多种硬件架构和操作系统的通用USB HOST驱动框架，能够在多个操作系统环境中无缝运行。
  + 自动加载机制：一个设计完善的二进制接口，支持驱动模块的按需动态加载，提高系统的灵活性和资源利用效率。
* 文档成果
  + 项目报告：详细描述项目的背景、意义、需求分析、系统设计和实现过程。
  + 使用手册：提供详细的用户手册，帮助用户理解和使用驱动框架，包括安装配置、基本操作和故障排除等。
  + 开发文档：详细记录开发过程中的技术细节，包括架构设计、模块接口、代码实现和测试方法等，便于后续维护和开发。
* 代码成果
  + GitHub仓库：在GitHub上开源项目代码，便于其他开发者查阅和使用。项目代码仓库链接为[https://github.com/dbydd/UUSBDriver](https://github.com/dbydd/UUSBDriver" \t "_new)。
  + 示范性Demo：提供一个完整的示范性Demo，展示驱动框架的基本功能和跨平台能力，包括基本的设备识别、数据传输和错误处理等。

这些成果将为后续的驱动开发和研究提供坚实的基础，并有助于推动跨操作系统USB驱动的标准化和推广应用。

# 4. 需求分析

## 4.1 用户需求分析

* 描述用户对跨操作系统USB驱动的需求，如高兼容性、易用性、高性能等。
* 用户场景和使用案例，如不同操作系统下的设备互操作需求。

## 4.2 系统需求分析

* 系统功能需求：详细描述系统应具备的功能，如设备识别、驱动加载、错误处理等。
* 性能需求：如响应时间、资源占用等方面的要求。
* 兼容性需求：支持的操作系统和硬件平台，如Windows、Linux、macOS，x86和ARM架构等。

# 5. 系统设计

## 5.1 关键技术

* **Rust编程语言**
  + **安全性**：Rust语言通过编译时的内存安全检查，避免了传统语言中的内存泄漏和非法内存访问问题，保证了驱动程序的安全性。
  + **性能**：Rust具有接近C/C++的高性能，能够满足驱动程序对性能的要求。
  + **可移植性**：Rust的跨平台特性使得驱动程序能够在多种操作系统和硬件架构上运行。
* **XHCI标准**
  + **通用性**：XHCI（eXtensible Host Controller Interface）是目前主流的USB HOST控制器标准，支持USB 1.x、2.0和3.x设备。
  + **技术细节**：实现XHCI控制器的驱动程序，支持设备的识别、初始化、数据传输和错误处理等功能。
* **模块化加载技术**
  + **二进制接口设计**：设计一个通用的二进制接口，支持驱动模块的动态加载和卸载，实现按需加载驱动程序的功能。
  + **动态链接库（DLL/Shared Library）**：使用动态链接库技术实现驱动模块的动态加载和调用，提高系统的灵活性和资源利用效率。

## 5.2 功能设计

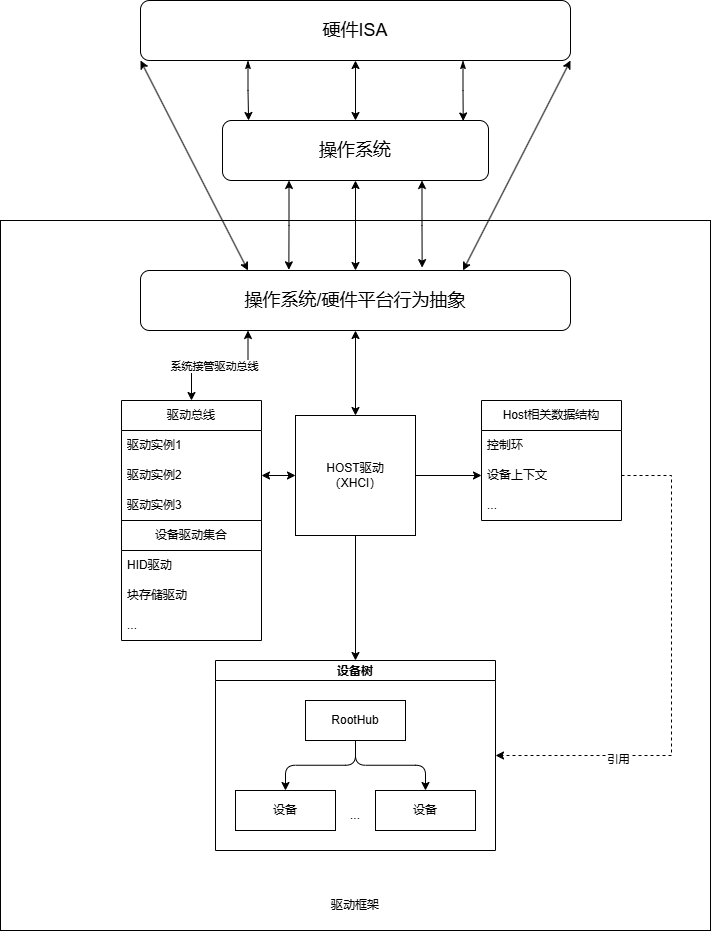
* **设备驱动集合**
  + 负责驱动程序的加载、卸载和管理，提供统一的接口

图 4-1 系统架构图

* + 支持驱动模块的按需加载，实现动态模块加载功能。
* **驱动总线**
  + 将驱动的运行过程转交给操作系统接管
  + 每个连接上的设备都对应一个驱动实例，驱动实例才是有使用功能的的数据对象
* **操作系统/硬件平台行为抽象**
  + 提供操作系统/硬件架构无关的抽象接口，将驱动程序与具体操作系统解耦，实现跨操作系统的兼容性。
  + 定义统一且精简的接口标准，简化驱动程序在不同操作系统间的移植和适配。
* **XHCI控制器模块**
  + 实现XHCI控制器的驱动功能，包括设备识别、初始化、数据传输和错误处理等。

通过以上设计，驱动框架能够实现跨操作系统和硬件架构的通用性和兼容性，并通过动态模块加载技术提高系统的灵活性和资源利用效率。

# 6. 参考文献

[1] H. Peng, M. Payer, USBFuzz: A Framework for Fuzzing USB Drivers by Device Emulation, in: 29th USENIX Security Symposium (USENIX Security 20), USENIX Association, 2020: pp. 2559–2575.

[2] J. Huang, Y. Wang, Research and Application of High-Speed and Adjustable Synchronous Data Transfer System Based on USB 3.0 Peripheral Controller, Journal of Circuits, Systems and Computers. 30 (2021) 2150118.

[3] N. Borgsmüller, The Rust programming language for embedded software development, phdthesis, Technische Hochschule Ingolstadt, 2021.

[4] 韩乃平,李蕾.国产操作系统生态体系建设现状分析[J].信息安全研究,2020,6(10):887-891.

[5] 杨志荣,胡茂海,董理,李守业.FPGA的xHCI协议软核IP设计及应用[J].单片机与嵌入式系统应用,2022,22(12):7-11.

[6] C. Wang, Analyzing the limitations of parallelism in hardware and software through threaded programming, in: Proceedings of 8th International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation (CDMMS 2023), 2023: pp. 24–29.

[7] 张帅林. 基于嵌入式处理器的高性能中间件软件设计与实现[D].北京理工大学,2021.DOI:10.26948/d.cnki.gbjlu.2018.001581.

[8] 刘巍,黄智勇.国产化实时通信中间件DDS的跨平台实现和优化技术[J].现代雷达,2019,41(07):39-43.DOI:10.16592/j.cnki.1004-7859.2019.07.007.

[9] 王硕,胡飞. 基于数据发布订阅服务的嵌入式通信中间件设计[C]//中国航空学会.第九届中国航空学会青年科技论坛论文集.中航出版传媒有限责任公司（China Aviation Publishing & Media CO.,2020:8.DOI:10.26914/c.cnkihy.2020.052267.

[10] 李荣.2022年物联网应用趋势[J].计算机与网络,2021,47(21):42-44.