**USB主机驱动技术的相关研究**

# 课题背景及意义

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，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。事实上，整个驱动开发领域，都有着诸多共性的问题。

## 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。H. Peng和M. Payer在29th USENIX Security Symposium上提出的USBFuzz[1]框架揭示了现有USB驱动中可能存在的安全漏洞，强调了对新开发驱动进行严格安全性测试的重要性。

## 软硬件协调问题

随着时代的发展，USB规范的更新，USB总线的吞吐容量越来越高，这也对相应的设备驱动，乃至于操作系统的性能提出了更高的要求，J. Huang和Y. Wang的研究[2]专注于基于USB 3.0外设控制器的高速同步数据传输系统，这表明未来USB驱动将更加注重性能优化。Rust语言是一种内存安全，提供零成本抽象的高级编程语言，其独特的设计理念与极高的性能优势使得其在面向计算机底层的开发的环境下大展身手，N. Borgsmüller的研究[3]指出，rust在嵌入式开发上比起传统的c语言开发在多个方面上更具优势。

## 操作系统依赖性强

2019年，图灵奖获得者John Hennessy和David Patterson在他们合著的《计算机体系结构：量化研究方法》中，提出了“计算机体系结构即将迎来新黄金年代”的观点。而我们也能看到，不同架构，设计理念的操作系统正如雨后春笋一般涌现出来。韩乃平和李蕾的文章[4]分析了国产操作系统的生态建设现状。对于一个成熟操作系统来说，USB系统的支持是不可或缺的，然而由于USB系统本身的复杂性，现有的USB驱动程序普遍都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。文章也指出，构建跨平台兼容的驱动程序对于推动技术自主可控具有重要意义

# 发展趋势

## 国外技术的发展情况

在国外，特别是在美国、欧洲以及日本等国家和地区，驱动开发领域已经发展得相当成熟。这些地区的科技公司和研究机构拥有悠久的历史，并且在硬件创新方面处于领先地位，这为驱动程序的开发提供了丰富的实践机会和技术积累。国外的技术生态体系中，开源文化非常盛行，例如Linux内核项目就汇集了来自全球各地贡献者的智慧，推动了包括驱动程序在内的各种软件组件的发展。此外，国际标准组织如USB-IF（USB Implementers Forum）和PCI-SIG（Peripheral Component Interconnect Special Interest Group）制定的标准促进了不同厂商产品之间的兼容性，使得驱动开发能够遵循统一规范进行。

许多大型跨国企业如Intel、NVIDIA、AMD等不仅自己开发高质量的驱动程序来支持自家产品，同时也积极参与到开放源代码社区中，分享其研究成果。这种合作模式加速了新技术的应用推广，比如针对最新处理器架构或图形处理单元（GPU）优化的驱动程序能够迅速被广大用户所采用。然而值得注意的是，此类大型实体所参与的开源往往是在现有的基础之上做的工作。举例来说，设备驱动往往是基于现有的应用广泛的操作系统而开发的，对于新的操作系统，存在着难以移植的问题。

## 国内相关技术的发展情况

近年来，中国在信息技术领域取得了显著进步，尤其是在移动互联网、大数据及人工智能等方面展现出了强劲的增长势头。然而，在驱动开发这一相对传统的IT分支上，国内起步较晚，整体水平仍存在一定差距。尽管如此，得益于政府对科技创新的支持力度加大以及市场环境的改善，本土企业在追赶国际先进水平的同时也逐渐形成了自己的特色。

一方面，以华为、中兴通讯为代表的通信设备制造商通过自主研发掌握了大量核心技术，其中包括适用于特定应用场景下的定制化驱动开发。另一方面，阿里巴巴、腾讯等互联网巨头也在积极布局底层基础设施建设，推出了基于云平台的服务端驱动解决方案，旨在为企业用户提供更高效稳定的数据处理能力。与此同时，一些专注于嵌入式系统研发的小型创业团队正努力填补国内市场空白，他们往往聚焦于某一垂直领域深入挖掘，力求提供差异化的产品和服务。除了大型企业之外，各家研究机构也加大了对于整个计算机体系结构方向的投入。而计算机体系结构是一个整体，具体的来说，软件与硬件之间有着联动的关系。随着国内硬件供应链的逐步完善，大量的硬件驱动开发需求也随之涌现，与此同时产生的效应是大量的开源社区成果的诞生。

# XHCI标准

XHCI（eXtensible Host Controller Interface）是目前主流的USB HOST控制器标准，其作为第三代USB主机控制器被设计出来，并向下兼容USB1.0/USB2.0规范。其设计使得上层USB驱动的部分与XHCI控制器本身的驱动解耦，提高了灵活性。同时XHCI从逻辑上是事件驱动的，对高IO，高并发的情况更为友好。杨志荣等人[5]介绍了FPGA的xHCI协议软核IP设计及其应用，为理解USB主机控制器接口提供了基础。C. Wang探讨了通过多线程编程分析硬件和软件并行性的局限性[6]，这对优化USB数据传输机制具有指导作用。张帅林在其博士论文中详细描述了基于嵌入式处理器的高性能中间件软件的设计与实现[7]，这对于提升USB驱动的整体性能至关重要。刘巍与黄智勇研究了DDS中间件的跨平台实现与优化技术[8]，为构建高效稳定的通信组件提供了参考。王硕与胡飞的工作则集中在基于数据发布订阅服务的嵌入式通信中间件设计上[9]，有助于探索事件驱动模型在USB驱动中的应用。此外，李荣总结了物联网领域的最新趋势[10]，强调了USB驱动作为核心基础设施的角色，特别是在日益增长的物联网设备互联场景下。

# RUST编程语言

Rust 是一种系统编程语言，专注于速度、内存安全和并行性。它由 Mozilla 研究院于2010年开始开发，并在2015年首次发布了1.0版本。Rust 的设计目标是提供与 C 和 C++ 类似的控制力，同时通过其所有权模型来保证内存安全，避免了常见的如空指针异常、数据竞争等错误。这种独特的所有权系统加上借用检查器使得 Rust 能够在编译时就捕捉到许多潜在的运行时错误，从而提高了软件的可靠性。

Rust 的语法简洁而富有表现力，支持函数式编程特性，例如闭包和迭代器。此外，它还提供了模式匹配、类型推断等功能，以简化代码编写过程。Rust 社区非常活跃，贡献了大量的库（称为 crates），这些库覆盖了从网络到图形处理等多个领域，为开发者提供了丰富的资源。

性能方面，Rust 生成的机器码可以直接运行且效率极高，这得益于其零成本抽象的原则——即所有抽象都应尽可能地不增加运行时开销。这意味着 Rust 可以被用来构建高性能的服务端应用、游戏引擎甚至是操作系统组件。

Rust 也注重跨平台兼容性，官方工具链支持多种操作系统和架构，允许开发者轻松地创建可以在不同环境下工作的软件。随着越来越多的企业和开源项目采用 Rust 来解决关键问题，它的生态系统正在迅速增长，成为现代软件工程中一个重要的选择。总之，对于追求高效能又不想牺牲安全性的开发者来说，Rust 提供了一个极具吸引力的选择。

# USB协议

USB（Universal Serial Bus，通用串行总线）是一种行业标准，用于定义电缆、连接器以及使用该接口的协议。自1996年首次推出以来，USB已经成为个人电脑及其外围设备之间最流行的连接方式之一。USB的设计目标是简化PC与外部设备之间的连接过程，使得用户可以轻松地添加或移除硬件而无需重启计算机，同时也减少了不同设备所需的多种类型的端口。

USB协议支持即插即用功能，这意味着当一个兼容USB的设备被连接到主机时，操作系统能够自动识别并配置该设备，从而允许其立即开始工作。USB最初版本1.0提供的最大传输速率为12Mbps，随后发展出了多个版本以适应不同的需求：USB 2.0将速度提升至480Mbps；USB 3.0进一步提高到了5Gbps，并且向后兼容旧版标准；最新的USB 4基于Thunderbolt 3技术，提供了高达40Gbps的数据传输速率，同时还支持更高功率的充电能力。

除了数据传输外，USB还能够为连接的设备供电，这对于移动存储设备、键盘鼠标等低功耗外设尤为重要。随着技术的进步，USB Power Delivery规范引入了更强大的电力输送能力，使得通过USB接口给笔记本电脑甚至显示器供电成为可能。

USB采用主从架构，其中一台设备作为主机控制整个通信过程，而其他设备则作为从属设备响应主机发出的请求。每个USB设备都拥有一个唯一的厂商ID和产品ID组合，这有助于操作系统正确识别并与之交互。为了确保良好的兼容性和互操作性，所有遵循USB标准的产品都需要经过认证测试。

随着时间的发展，USB不仅限于传统的A型和B型接口，还包括了Micro-USB、Mini-USB及Type-C等多种形态，特别是Type-C由于其小巧且正反面均可插入的特点，在近年来得到了广泛应用。总的来说，凭借其灵活性、便捷性和不断演进的技术规格，USB继续在全球范围内保持着重要地位，为消费者带来了极大的便利。

# POLYHAL（多架构HAL层）

HAL（Hardware Abstraction Layer，硬件抽象层）是位于操作系统或软件应用程序与物理硬件之间的一层软件接口。它的主要目的是为上层软件提供一种统一的方式来访问底层硬件资源，而无需直接处理各种不同硬件的具体细节。通过这种抽象化的方式，HAL帮助开发者编写更加通用、可移植性更强的代码，并简化了设备驱动程序的开发过程。

在计算机系统中，不同的硬件组件如处理器、存储器、输入输出设备等具有各自的特性和操作方式。如果没有HAL的存在，每当需要支持新的硬件时，操作系统或其他软件就必须进行相应的修改以适应这些变化。然而，有了HAL之后，只需要更新HAL本身或者相关的驱动程序即可，这大大降低了维护成本并提高了系统的灵活性。

HAL通常会实现一组标准的API（应用程序编程接口），这些API定义了如何执行常见的硬件相关任务，比如读写I/O端口、控制中断处理以及管理内存映射等。这样一来，即使底层硬件发生变化，只要它们能够通过相同的API被调用，那么对于上层的应用来说就几乎是透明的。

此外，HAL还有助于提高安全性，因为它可以在一定程度上限制对硬件资源的直接访问，从而减少因错误操作而导致的问题。例如，在某些安全敏感的应用场景下，可能不允许直接访问某些关键硬件部件；此时就可以通过HAL来实施必要的权限检查和隔离措施。

POLYHAL的想法源自于清华大学软件所的陈渝教授的一篇博客，其目的是为后续的模块化操作系统生态提供一份扎实的基础。目前POLYHAL使用Rust编程语言编写，在Github上开源。其已支持riscv/x86/aarch64/龙芯等多种架构，提供了中断/时钟/页表/虚拟地址/TLB等CPU ISA基础设施的操作抽象。

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.