Embedded Technoloy and Intelligent Systems 嵌入式技术与智能系统, 2025, 2(4), 213-222
Published Online August 2025 in Hans. https://www.hanspub.org/journal/etis
https://doi.org/10.12677/etis.2025.24019

嵌入式操作系统技术发展趋势

张　润，夏　晨，陈咸彰

重庆大学计算机学院，重庆

收稿日期：2025年10月28日；录用日期：2025年11月20日；发布日期：2025年12月1日

摘　要

随着信息技术的迅速发展，嵌入式系统已由传统的单一功能控制器演进为支撑物联网、工业互联网、智能汽车和消费电子等领域的核心计算平台。嵌入式操作系统(Embedded Operating System, EOS)作为嵌入式设备的基础软件，在资源调度、安全隔离和智能协同方面发挥着关键作用，其性能、可靠性与安全性直接影响系统的实时性、能效与可信度。文章系统梳理了嵌入式操作系统的研究现状与发展趋势。从安全架构、漏洞防御、模糊测试、形式化验证等方面总结了国内外在嵌入式系统安全技术方面的主要进展；从调度优化、能效管理、内存与缓存协同设计等角度分析了性能优化的关键技术；在文件系统层面，归纳了面向资源受限设备的高效文件管理与闪存优化方案；最后，从智能化、泛在化及基于Rust语言的安全可扩展方向，探讨了嵌入式操作系统未来的发展趋势。研究表明，嵌入式操作系统正由传统的“轻量与实时”逐步迈向“安全、智能与泛在”的新阶段，成为万物互联时代的关键基础软件。

关键词

嵌入式操作系统，物联网，工业自动化

Development Trends in Embedded Operating System Technology

Run Zhang, Chen Xia, Xianzhang Chen

College of Computer Science, Chongqing University, Chongqing

Received: October 28, 2025; accepted: November 20, 2025; published: December 1, 2025

Abstract

With the rapid advancement of information technology, embedded systems have evolved from traditional single-function controllers into the core computing platforms that support the Internet of Things (IoT), industrial Internet, smart vehicles, and consumer electronics. As the fundamental software of embedded devices, the Embedded Operating System (EOS) plays a crucial role in resource

文章引用: 张润, 夏晨, 陈咸彰. 嵌入式操作系统技术发展趋势[J]. 嵌入式技术与智能系统, 2025, 2(4): 213-222.
DOI: 10.12677/etis.2025.24019

scheduling, security isolation, and intelligent collaboration. Its performance, reliability, and security directly determine the system's real-time responsiveness, energy efficiency, and trustworthiness. This paper systematically reviews the current research status and future trends of embedded operating systems. It summarizes recent progress in embedded system security technologies—including system architecture design, vulnerability defense, fuzz testing, and formal verification. From the perspective of performance, it analyzes key techniques such as scheduling optimization, energy-efficient management, and cache-algorithm co-design. In the domain of file systems, it explores efficient file management and flash memory optimization strategies for resource-constrained devices. Finally, the paper discusses future development directions toward intelligent, ubiquitous, and Rust-based secure embedded operating systems. The study indicates that embedded operating systems are transitioning from the traditional focus on "lightweight and real-time" features to a new era characterized by "security, intelligence, and ubiquity," becoming a cornerstone of the Internet of Everything.

Keywords

Embedded Operating System, Internet of Things, Industrial Automation

1. 引言

随着信息技术的不断进步，嵌入式系统已逐渐从早期的单一功能控制器，发展成为支撑物联网、工业互联网、智能汽车与消费电子的核心计算平台。在这一转型过程中，嵌入式操作系统(Embedded Operating System, EOS)作为嵌入式设备的基础软件，扮演着资源调度、安全隔离和智能协同的关键角色。其性能与可靠性不仅直接决定设备的实时性和能效表现，更在安全可信与智能化应用中发挥着基础性作用。

近年来，嵌入式操作系统发展出了安全关键嵌入式操作系统、物联网嵌入式操作系统、智能嵌入式操作系统和泛在嵌入式操作系统等类型。图 1 展示了嵌入式操作系统的四大分类及其代表性系统的发展时间线，其中，安全关键/分区嵌入式操作系统(如 INTEGRITY、VxWorks 653)专注于高可靠性和任务隔离，适用于航空航天和工业控制等领域。物联网嵌入式操作系统(如 FreeRTOS、Zephyr)以轻量化和多协议支持为核心，广泛应用于智能家居和工业物联网。智能嵌入式操作系统(如 LiteOS、Google Coral)集成了 AI 本地化推理能力，支持设备端智能决策。泛在嵌入式操作系统(如 HarmonyOS、XIUOS)则强调跨设备协同和生态统一性，推动全场景智能互联。随着技术的融合与发展，嵌入式操作系统正逐步向智能化、泛在化演进，成为万物互联时代的核心基础软件。本文将介绍支撑四类嵌入式操作系统发展的关键技术进展。

2. 安全技术

嵌入式操作系统广泛应用于航空航天、工业控制、智能交通等关键领域，其安全性直接关系到系统的稳定运行和数据安全。随着物联网和人工智能技术的发展，嵌入式系统面临的安全威胁日益复杂多样，如漏洞利用、数据泄露、恶意攻击等。因此，保障嵌入式操作系统的信息安全成为当前研究的热点和难点。本节从系统安全架构与机制、安全管理机制、漏洞模糊测试、安全检测与验证等方面，对近年来国

内外关于嵌入式操作系统信息安全的研究进行了分类总结，如图 2 所示。

Figure 1. Timeline of embedded operating system development
图 1. 嵌入式操作系统发展时间线

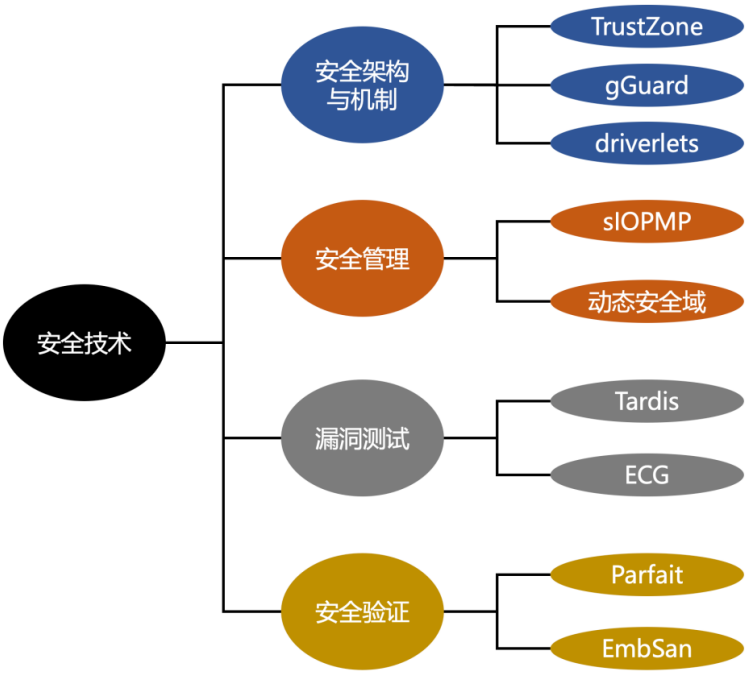

Figure 2. Classification of security technologies for embedded operating systems
图 2. 嵌入式操作系统安全技术分类

(1) 安全架构与机制

嵌入式操作系统的安全机制与架构设计是防止漏洞被利用的关键。一个健壮的安全架构可以有效隔离潜在的攻击面，限制恶意行为的扩散。当前，ARM Cortex-M TrustZone 等技术被广泛应用于嵌入式系

统中，以实现安全区域的划分和保护。然而，这些技术在实际应用中仍存在一些安全隐患和性能瓶颈。例如，快速状态切换机制可能导致安全状态和非安全状态之间的权限控制出现漏洞，从而被攻击者利用。因此，研究更精细、高效的安全机制与架构，对于提升嵌入式系统的整体安全性具有重要意义。

现有实时系统越来越多地使用现成的组件，并且经常连接到互联网，从而暴露于各种攻击之下。Hasan等人[1]提出了“攻击者负担”(attacker’s burden)这一度量指标，用于评估实时系统中基于调度器的安全机制的有效性。该指标基于“工作量因子”(work factor)的概念，通过计算攻击者在不同安全机制下可利用的时间来衡量安全性的强弱。具体来说，攻击者负担越高，意味着攻击者可利用的时间越少，系统的安全性越高。Yadlapall 等人[2]提出的 gGuard 通过在编译器、库和操作系统层面实现高效的数据粉碎技术，防止信息泄露。具体来说，gGuard 在编译时分析应用程序的数据访问模式和依赖关系，将原始程序转换为“自粉碎”版本，在执行过程中自动清理不再使用的内存对象。对于无法在执行期间清理的内存对象，gGuard 采用异步清理方法，减少清理操作对性能的影响。ARM Cortex-M TrustZone 技术是 ARM 公司推出的硬件安全扩展，旨在为嵌入式系统提供基于硬件的安全隔离机制。其核心思想是通过将处理器的执行环境划分为安全世界(Secure World)和非安全世界(Non-Secure World)，实现对密钥、用户数据、安全固件等敏感资源的隔离保护。Ma 等人[3]针对 ARM Cortex-M TrustZone 架构中快速状态切换机制的安全隐患，提出了一种名为 return-to-non-secure 的攻击技术，并设计了两种地址清理机制进行防御，实现了在不影响系统性能的前提下，有效防止特权升级和任意代码执行的效果。Park 等人[4]针对在移动设备的 TrustZone 可信执行环境中运行敏感 GPU 计算时，如何安全且高效地部署 GPU 软件栈的问题，提出了 GR-T 架构，通过移动设备与无 GPU 的云服务协作进行 GPU 记录与回放，实现了安全、高效的 GPU 计算，同时显著降低了客户端的时间和能量消耗。Kang 等人[5]针对嵌入式系统中广泛使用的内存不安全 C/C++代码导致的堆内存安全问题，提出了基于 ARMv8-M 架构的微胖指针方案，实现了一种高效的边界检查机制，确保嵌入式系统的堆内存安全，同时性能开销较低。Guo 等人[6]针对 ARM TrustZone 缺乏现代 I/O 设备驱动程序的问题，提出了一种称为“driverlets”的新型方法。通过记录和回放全功能驱动程序与设备的交互过程，生成最小可行的驱动程序，解决了 TrustZone 中安全 I/O 的关键挑战。

(2) 安全管理机制

高沙沙等人[7]针对现有基于 MILS 架构的嵌入式操作系统在任务故障后无法实现功能重构和实时动态加载的问题，提出了面向任务的多级安全域动态管理架构。通过安全域管理、故障监控和功能重构模块，实现任务在特定安全域内的动态迁移与功能恢复。确保了任务在故障后的持续运行，提升了系统的可靠性和安全性。Feng 等人[8]针对现有 I/O 隔离机制在 TEE 系统中存在性能和可扩展性问题，提出了 sIOPMP 机制。通过多级树形检查器、可挂载 IOPMP 和 IOPMP 重映射等设计，实现了高效且可扩展的 I/O 保护。多级树形检查器支持超过 1000 个硬件区域，可挂载 IOPMP 支持无限数量的设备，而 IOPMP 重映射机制则允许设备在热和冷状态之间动态切换，以适应不同的 I/O 工作负载。Guo 等人[9]针对大规模关键任务 IoT 系统的可扩展性、安全性和实时性挑战，提出跨层解决方案，涵盖数据驱动建模、动态资源管理、形式化漏洞检测和自适应传感器攻击防御。通过层次化控制架构和分布式资源管理，提升了系统的可扩展性、安全性及对物理攻击的实时响应能力。

(3) 漏洞模糊测试

嵌入式操作系统的漏洞可能导致系统崩溃、数据泄露甚至恶意攻击，严重威胁关键基础设施的安全。传统的测试方法难以全面、高效地发现这些漏洞，因此，漏洞检测与模糊测试技术成为保障嵌入式系统安全的重要手段[10]。早期的黑盒测试方法虽然简单易行，但效率低下，难以覆盖复杂的代码路径。随着技术发展，覆盖引导模糊测试等方法逐渐兴起，但仍存在对特定架构依赖强、生成测试用例质量不高等问题。Shen 等人[11]提出的 Tardis 是一种覆盖引导的嵌入式操作系统模糊测试工具。它通过在编译时对

目标操作系统进行插桩，实现与操作系统无关的代码覆盖收集和分析。具体来说，Tardis 使用基于位图的存储方式记录每个执行的代码分支，并通过共享内存缓冲区将覆盖信息传递给主机端的模糊测试引擎。主机端的分析模块利用简单的位运算快速判断测试用例是否触发了新的代码路径，从而指导后续的测试用例生成和变异。Zhang 等人[12]提出的 ECG 利用大型语言模型(LLM)来增强嵌入式操作系统的模糊测试能力。ECG 首先通过静态分析提取目标系统调用的参数类型和约束信息，并结合文档生成实际的 C 代码示例。然后，利用 strace 捕获执行跟踪信息，将其转换为系统调用规范。在模糊测试过程中，ECG 使用 LLM 生成测试用例，并根据执行反馈动态调整生成策略，以提高测试用例的质量和覆盖范围。

(4) 安全检测与验证

如何有效检测并修复嵌入式系统中的潜在漏洞，以及如何从形式上验证其安全性，仍然是嵌入式系统领域的热点问题。Liu 等人[13]针对嵌入式操作系统固件缺乏有效漏洞检测手段的问题，提出 EmbSan 框架，利用动态插桩和解耦的主机运行时库实现对多种嵌入式操作系统固件的漏洞检测，成功检测出实际漏洞，且性能开销与现有内核检测工具相当，提升了检测效率与适应性。Athalye 等人[14]针对硬件安全模块(HSM)从应用规范到电路实现的安全性和无泄漏验证问题，提出 Parfait 框架，通过信息保持精化(IPR)的形式化方法，利用中间抽象层次和传递性信息保持精化实现模块化验证，成功验证了四个 HSM 实例的安全性，确保其无正确性、安全性和漏洞泄漏，且具有良好的可扩展性和适应性。Pasquier 等人[15]针对嵌入式操作系统安全分析工具选择困难的问题，提出通过构建设备驱动案例研究并应用多种分析技术(如模型检查、代码分析等)来评估工具适用性，实现了对嵌入式操作系统模型和代码分析工具的部分互补性分析，为工具选择提供依据。

3. 调度优化

嵌入式系统由于资源受限、能耗敏感与实时性要求高，其调度机制需在性能、功耗与任务时限之间实现最优平衡。针对这一需求，研究者提出了多种资源与任务协同优化策略，如图 3 所示。

Figure 3. Scheduling optimization in embedded operating systems
图 3. 嵌入式操作系统中的调度优化

Mamata 等人[16]提出了一种面向资源优化的调度模型，以最小化实时嵌入式系统中的资源过度配置。该方法利用供需约束函数优化资源分配，在保证任务时限的同时，平均减少了 13%的资源使用量。Reghenzani 等人[17]提出了一种多级动态电源管理(DPM)策略，以优化异构处理器上实时有向无环图任务的能耗。该方法结合任务概率执行时间分析，动态调整处理器 C-States，在保证硬实时约束的前提下，实现了最高 32.1%的能耗降低，并通过仿真和实测验证了优化效果。Li 等人[18]提出了一种协作式内存和进程管理框架 ICE。ICE 通过在内存页面故障时冻结后台进程，并基于内存压力动态解冻，从而有效减少后台应用引发的内存故障，提升了 1.57 倍的帧率和减少了交互警告。Jiang 等人[19]提出了一种面向并行实时系统的缓存与算法协同设计方案。该研究引入了一种虚拟索引、物理标记、选择性包含的 L1.5 缓存，

并提出了针对有向无环图任务的调度方法，以减少数据传输延迟并降低任务完工时间。Wu 等人[20]提出了一种预测性共享最后级缓存(LLC)分区的方法，方法采用了 1S-TDM 调度策略并结合硬件扩展集合序列器，显著降低了最坏情况延迟(WCL)，提高了共享缓存分区的利用效率。Chen 等人[21]提出了一种基于细粒度共享资源模型的改进阻塞分析方法，解决并行实时任务中非抢占式自旋锁的阻塞问题。方法通过线性优化技术实现了无膨胀分析，避免了传统分析中的过度表示。

4. 文件系统

嵌入式设备具有存储资源受限、计算能力有限、成本敏感等特点，导致嵌入式系统在文件系统设计方面面临重大挑战。现有研究从空间管理粒度与元数据精简、数据结构与页分配优化、压缩/缓存以抑制读写放大、以及高效垃圾回收策略等方向展开，如图 4 所示。

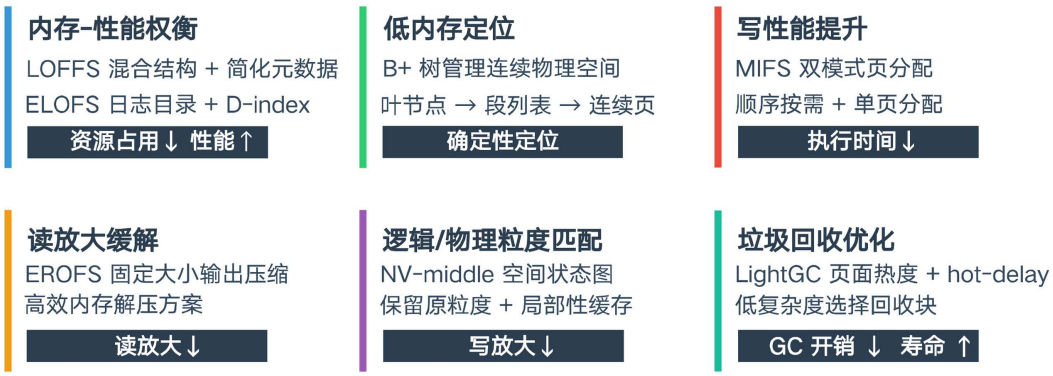

Figure 4. File system optimization techniques in embedded operating systems
图 4. 嵌入式操作系统中的文件系统优化技术

为此，Zhang 等人[22]提出了一种新型嵌入式文件系统 LOFFS，以解决嵌入式文件系统由于内存消耗过多和启动功能差而无法有效管理大容量 NAND 闪存的问题。LOFFS 重新定义了空间管理粒度并简化元数据以加快挂载程序，为不同类型的文件提出了混合文件结构，以解决性能和内存占用之间的权衡，从而以最小的资源占用实现高性能。Jiang 等人[23]为减少数据传输延迟及内存占用，采用 B+树来管理连续物理空间，令每个叶节点指向一个段列表，引用一组物理上连续的页面。确保在低内存占用的情况下实现确定性的文件数据定位。为了提升写操作性能，MIFS 提出了包括顺序按需分配以及单页分配的双模式页分配策略，将根据不同的情况采用不同的模式以减少执行时间。Gao 等人[24]针对嵌入式系统的有限存储空间和有限运行时内存，分析了现有压缩只读文件系统中的 I/O 放大和冗余计算的问题，认为传统压缩只读文件系统的问题在于采用了固定大小的输入压缩策略。因此，EROFS 引入了固定大小的输出压缩策略，并设计了高效记忆的解压方案，以缓解读放大率问题并减少不必要的计算。

闪存采用页和块的粒度对物理空间进行管理，大于现有嵌入式文件系统的逻辑管理单位。因此，将嵌入式文件系统移植到闪存介质之上将导致逻辑 I/O 粒度和物理 I/O 粒度不匹配的问题，手动扩大文件系统系统的逻辑 I/O 粒度将带来大量的移植工作，也将产生写放大问题，导致额外的空间消耗和性能降低。Zhang 等人[25]提出了存储中间件 NV-middle，保留了嵌入式文件系统原有的操作粒度，定义了空间状态图来标识属于同一物理块的逻辑块的状态，并根据逻辑/物理页面与块之间的关系以及数据访问的局部性，优化了内存使用策略和页面缓存机制。此外，Zhang 等人[26]提出了一种新型嵌入式文件系统 ELOFS，重新定义了空间管理粒度并简化了元数据以提升性能。同时根据目录和数据文件不同的访问模式设计了混合文件结构，以日志方式组织频繁更新的目录文件，将数据页分散在闪存上，并通过 D-index

进行索引。Sun 等人[27]发现闪存垃圾回收的开销占总 I/O 时间的 85%以上，降低了嵌入式文件系统的性能，并缩短了闪存芯片的耐用性，而删除不同热度的数据块的影响也有所差异。针对上述问题，该文献提出了低复杂度、高效率的 GC 方案 LightGC，基于页面更新间隔来衡量页面的热度，并使用 hot-delay 策略衡量擦除块在未来一段时间内的影响，以选择垃圾回收的数据块。

5. 发展趋势

人机物融合智能泛在操作系统持续发展：梅宏院士等[28]探讨了面向人机物融合泛在计算的泛在操作系统，认为泛在操作系统是操作系统发展的新蓝海。在此概念下，北京大学发布了一款面向工业物联场景的泛在操作系统——XiUOS 矽璡工业物联操作系统[29]。Syswonder 社区正推动在不同应用场景中的泛在操作系统的构建及技术应用，开发了基于 Rust 语言开发的 Unikernel 操作系统以及基于硬件分区的 Hypervisor [30]。

随着人、机、物的深度互联融合，泛在操作系统正由“场景适配”向“场景自适应”方向演进。未来的泛在操作系统将进一步实现跨平台、跨终端的智能协同，支持多态计算与实时资源编排。同时，可信计算与边缘智能将成为推动泛在操作系统落地的关键力量。未来，泛在操作系统有望在工业互联网、智慧城市和车联网等场景中形成生态级应用集群，构建以操作系统为核心的智能底座。

AI 原生嵌入式操作系统：学术界着重研究高效、轻量化的智能嵌入式操作系统。Luo 等人[31]提出了首个由深至浅的可转换神经架构搜索(NAS)范式——Double-Win NAS (DW-NAS)，探索兼具深层网络高精度和浅层网络高效硬件性能的网络架构。Pasricha 等人[32]探讨了在物联网平台上部署机器学习软件的挑战，提出了简化部署的策略。Van Delm 等人[33]提出了 HTVM 编译器，最大限度地提高异构加速器的利用率并减少数据移动。Han 等人[34]提出了 DTMM 库，旨在解决在物联网设备上高效部署和执行机器学习模型的挑战。Xu 等人[35]提出了一种超轻量级神经网络，专为资源受限的低功耗设备设计。

AI 原生嵌入式操作系统正逐步由上述的融合 AI 模式迈向 AI 内生模式。未来系统将配备以 AI 推理引擎、模型/智能体调度器等构成的 AIOS 内核，形成 AIOS 内核与传统内核的深度共生架构。AI 原生 OS 不仅将赋能边缘端智能感知、决策与自优化能力，还将推动软硬件协同设计的新范式。随着 TinyML、边缘大模型与自适应能耗管理技术的发展，AI 原生操作系统有望成为物联网时代智能计算的关键支撑层。

Rust 成为热门内核开发语言：Rust 语言凭借其内存安全和类型安全的特性，逐渐成为嵌入式操作系统开发的热点工具。Li 等人[36]对 Rust-for-Linux 项目进行了首次实证研究，分析了 Rust 如何与 Linux 内核融合以及其在安全性和性能方面的表现。Hu 等人[37]提出一款基于 Rust 的高性能嵌入式单核内核，利用 Rust 的内存安全特性减少错误并确保可靠的资源管理。Narayanan 等人[38]开发了一个基于 Rust 开发的新型操作系统，通过语言级别的类型和内存安全实现轻量级细粒度隔离。Ma 等人[39]探讨了在基于 Rust 的软实时嵌入式操作系统中实现堆栈展开和恐慌恢复的可行性，针对资源受限环境提出了几种新颖的优化方法。中关村实验室联合发布的“星绽 OS”采用 Rust 语言构建框内核架构，最大限度减少非安全代码的比例，解决了传统 C 语言在内存安全方面的潜在问题。此外，Rust 在嵌入式操作系统领域的发展也存在一些问题，如 Sharma 等人[40]系统性研究了 Rust 在嵌入式系统中的应用现状和挑战，研究表明 Rust 需要进一步改进工具支持、增强与现有代码库的互操作性，并为开发人员提供更好的资源和支持，以推动 Rust 在嵌入式领域的普及。Ayers 等人[41]研究了 Rust 在嵌入式系统中二进制文件过大的问题，提出了减少二进制文件大小的编程原则。

随着 Rust-for-Linux 项目成熟及产业生态完善，Rust 有望成为下一代操作系统内核的主流开发语言。未来的发展重点将聚焦于工具链完备性、实时性优化及与 C/C++生态的无缝互操作。同时，Rust 在安全物联网、车载控制系统及国防级实时系统中的应用将进一步扩展，形成安全可信且高性能的系统开发新范式。

6. 结语

随着物联网、人工智能、工业自动化等技术的迅猛发展，嵌入式操作系统正从传统的“轻量、实时”向“安全、智能、泛在”方向演进，成为支撑未来智能互联世界的关键基础软件。本文系统梳理了嵌入式操作系统在安全架构、调度优化、文件系统等方面的研究进展，并探讨了智能化、泛在化及基于 Rust 语言的安全开发趋势。

未来的嵌入式操作系统将面临更加复杂的应用场景与更高的性能要求，不仅要保障系统的实时性与稳定性，还需应对多样化的安全威胁与资源受限的挑战。因此，发展具备自适应能力、智能协同机制与高可信保障的新型嵌入式操作系统，将是学术界与产业界共同努力的方向。

嵌入式操作系统的发展不仅是技术演进的体现，更是推动数字社会、智能产业和绿色计算的重要力量。随着新架构、新语言和新模型的不断涌现，嵌入式操作系统将在万物互联时代扮演更加核心的角色，助力构建更加安全、智能与可持续的计算生态。

基金项目

国家自然科学基金面上项目(62372073)，面向新型可计算存储的高效任务管理及调度研究，2024.1~2027.12。

参考文献

[1] Hasan, M., Kashinath, A., Chen, C. and Mohan, S. (2024) Sok: Security in Real-Time Systems. *ACM Computing Surveys*, **56**, 1-31. https://doi.org/10.1145/3649499

[2] Yadlapalli, Y., Zhou, H., Zhang, Y. and Liu, C. (2021) gGuard: Enabling Leakage-Resilient Memory Isolation in GPU-Accelerated Autonomous Embedded Systems. 2021 58*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 5-9 December 2021, 817-822. https://doi.org/10.1109/dac18074.2021.9586244

[3] Ma, Z., Tan, X., Ziarek, L., Zhang, N., Hu, H. and Zhao, Z. (2023) Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense. 2023 60*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 9-13 July 2023, 1-6. https://doi.org/10.1109/dac56929.2023.10247972

[4] Park, H. and Lin, F.X. (2023) Safe and Practical GPU Computation in TrustZone. *Proceedings of the Eighteenth European Conference on Computer Systems*, Rome, 8-12 May 2023, 505-520. https://doi.org/10.1145/3552326.3567483

[5] Kang, J., Park, J., Seo, J. and Kwon, D. (2024) Look before You Access: Efficient Heap Memory Safety for Embedded Systems on ARMv8-M. *Proceedings of the* 61*st ACM/IEEE Design Automation Conference*, San Francisco, 23-27 June 2024, 1-6. https://doi.org/10.1145/3649329.3655949

[6] Guo, L. and Lin, F.X. (2022) Minimum Viable Device Drivers for ARM TrustZone. *Proceedings of the Seventeenth European Conference on Computer Systems*, Rennes, 5-8 April 2022, 300-316. https://doi.org/10.1145/3492321.3519565

[7] 高沙沙, 王中华. 基于MILS架构的嵌入式操作系统多级安全域动态管理技术[J]. 计算机科学, 2019, 46(S2): 460-463.

[8] Feng, E., Feng, D., Du, D., Xia, Y., Zheng, W., Zhao, S., *et al.* (2024) SIOPMP: Scalable and Efficient I/O Protection for Tees. *Proceedings of the* 29*th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Volume 2, La Jolla, 27 April 2024-1 May 2024, 1061-1076. https://doi.org/10.1145/3620665.3640378

[9] Guo, X., Han, S., Hu, X.S., Jiao, X., Jin, Y., Kong, F., *et al.* (2021) Towards Scalable, Secure, and Smart Mission-Critical IoT Systems. *Proceedings of the* 2021 *International Conference on Embedded Software*, Virtual, 8-15, 2021 October 1-10. https://doi.org/10.1145/3477244.3477624

[10] Yun, J., Rustamov, F., Kim, J. and Shin, Y. (2022) Fuzzing of Embedded Systems: A Survey. *ACM Computing Surveys*, **55**, 1-33. https://doi.org/10.1145/3538644

[11] Shen, Y., Xu, Y., Sun, H., Liu, J., Xu, Z., Cui, A., *et al.* (2022) Tardis: Coverage-Guided Embedded Operating System Fuzzing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **41**, 4563-4574. https://doi.org/10.1109/tcad.2022.3198910

[12] Zhang, Q., Shen, Y., Liu, J., Xu, Y., Shi, H., Jiang, Y., *et al.* (2024) *ECG*: Augmenting Embedded Operating System

Fuzzing via LLM-Based Corpus Generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **43**, 4238-4249. https://doi.org/10.1109/tcad.2024.3447220

[13] Liu, J., Shen, Y., Xu, Y., Sun, H., Shi, H. and Jiang, Y. (2024) Effectively Sanitizing Embedded Operating Systems. *Proceedings of the* 61*st ACM/IEEE Design Automation Conference*, San Francisco, 23-27 June 2024, 1-6. https://doi.org/10.1145/3649329.3658490

[14] Athalye, A., Corrigan-Gibbs, H., Kaashoek, F., Tassarotti, J. and Zeldovich, N. (2024) Modular Verification of Secure and Leakage-Free Systems: From Application Specification to Circuit-Level Implementation. *Proceedings of the ACM SIGOPS* 30*th Symposium on Operating Systems Principles*, Austin, 4-6 November 2024, 655-672. https://doi.org/10.1145/3694715.3695956

[15] Pasquier, M., Jouault, F., Brun, M. and Pérochon, J. (2020) Evaluating Tool Support for Embedded Operating System Security. *Proceedings of the* 23*rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*: *Companion Proceedings*, Virtual, 16-23 October 2020, 1-10. https://doi.org/10.1145/3417990.3420048

[16] Mamata, R. and Azim, A. (2022) Work-in-Progress: A Resource-Aware Optimization Model for Real-Time Systems Analysis and Design. 2022 *International Conference on Embedded Software* (*EMSOFT*), Shanghai, 7-14 October 2022, 9-10. https://doi.org/10.1109/emsoft55006.2022.00012

[17] Reghenzani, F., Bhuiyan, A., Fornaciari, W. and Guo, Z. (2021) A Multi-Level DPM Approach for Real-Time DAG Tasks in Heterogeneous Processors. 2021 *IEEE Real-Time Systems Symposium* (*RTSS*), Dortmund, 7 December 2021, 14-26. https://doi.org/10.1109/rtss52674.2021.00014

[18] Li, C., Liang, Y., Ausavarungnirun, R., Zhu, Z., Shi, L. and Xue, C.J. (2023) ICE: Collaborating Memory and Process Management for User Experience on Resource-Limited Mobile Devices. *Proceedings of the Eighteenth European Conference on Computer Systems*, Rome, 8-12 May 2023, 79-93. https://doi.org/10.1145/3552326.3567497

[19] Jiang, Z., Zhao, S., Wei, R., Gao, Y. and Li, J. (2024) A Cache/Algorithm Co-Design for Parallel Real-Time Systems with Data Dependency on Multi/Many-Core System-on-Chips. *Proceedings of the* 61*st ACM/IEEE Design Automation Conference*, San Francisco, 23-27 June 2024, 1-6. https://doi.org/10.1145/3649329.3657372

[20] Wu, Z. and Patel, H. (2022) Predictable Sharing of Last-Level Cache Partitions for Multi-Core Safety-Critical Systems. *Proceedings of the* 59*th ACM/IEEE Design Automation Conference*, San Francisco, 10-14 July 2022, 1273-1278. https://doi.org/10.1145/3489517.3530614

[21] Chen, Z., Lei, H., Yang, M., Liao, Y. and Qiao, L. (2021) A Finer-Grained Blocking Analysis for Parallel Real-Time Tasks with Spin-Locks. 2021 58*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 5-9 December 2021, 1177-1182. https://doi.org/10.1109/dac18074.2021.9586270

[22] Zhang, R., Liu, D., Chen, X., She, X., Yang, C., Tan, Y., *et al.* (2020) LOFFS: A Low-Overhead File System for Large Flash Memory on Embedded Devices. 2020 57*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 20-24 July 2020, 1-6. https://doi.org/10.1109/dac18072.2020.9218635

[23] Jiang, J., Yang, M., Qiao, L., Wang, T. and Chen, X. (2025) MIFS: A Low Overhead and Efficient Mixture File Index Management Method in Flash File System. *Journal of Systems Architecture*, **162**, Article 103387. https://doi.org/10.1016/j.sysarc.2025.103387

[24] Gao, X., Dong, M., Miao, X., *et al.* (2019) EROFS: A Compression-Friendly Readonly File System for Resource-Scarce Devices. 2019 *USENIX Annual Technical Conference*, Washington, 10-12 July 2019, 149-162.

[25] Zhang, R., Liu, D., Shen, Z., She, X., Yang, C., Chen, X., *et al.* (2021) Bridging Mismatched Granularity between Embedded File Systems and Flash Memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **40**, 2024-2035. https://doi.org/10.1109/tcad.2020.3036814

[26] Zhang, R., Liu, D., Chen, X., She, X., Yang, C., Tan, Y., *et al.* (2022) ELOFS: An Extensible Low-Overhead Flash File System for Resource-Scarce Embedded Devices. *IEEE Transactions on Computers*, **71**, 2327-2340. https://doi.org/10.1109/tc.2022.3152079

[27] Sun, D., Song, Y., Chai, Y., Peng, B., Lu, F. and Deng, X. (2022) Light-GC. *Proceedings of the* 23*rd ACM/IFIP International Middleware Conference*, New York, 7-11 November 2022, 216-227. https://doi.org/10.1145/3528535.3565246

[28] 梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融合泛在计算的新蓝海[J]. 中国科学院院刊, 2022, 37(1): 30-37.

[29] Cao, D., Xue, D., Ma, Z. and Mei, H. (2022) XiUOS: An Open-Source Ubiquitous Operating System for Industrial Internet of Things. *Science China Information Sciences*, **65**, Article 117101. https://doi.org/10.1007/s11432-021-3294-y

[30] RuxOS 手册[Z/OL]. https://ruxos.syswonder.org/, 2025-09-15.

[31] Luo, X., Liu, D., Kong, H., Huai, S. and Liu, W. (2024) Double-Win NAS: Towards Deep-to-Shallow Transformable Neural Architecture Search for Intelligent Embedded Systems. *Proceedings of the* 61*st ACM/IEEE Design Automation Conference*, San Francisco, 23-27 June 2024, 1-6. https://doi.org/10.1145/3649329.3656247

[32] Pasricha, S. (2023) Lightning Talk: Efficient Embedded Machine Learning Deployment on Edge and IoT Devices. 2023

60*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 9-13 July 2023, 1-2. https://doi.org/10.1109/dac56929.2023.10247845

[33] Van Delm, J., Vandersteegen, M., Burrello, A., Sarda, G.M., Conti, F., Pagliari, D.J., *et al.* (2023) HTVM: Efficient Neural Network Deployment on Heterogeneous TinyML Platforms. 2023 60*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 9-13 July 2023,1-6. https://doi.org/10.1109/dac56929.2023.10247664

[34] Han, L., Xiao, Z. and Li, Z. (2024) DTMM: Deploying TinyML Models on Extremely Weak IoT Devices with Pruning. *IEEE INFOCOM* 2024-*IEEE Conference on Computer Communications*, Vancouver, 20-23 May 2024, 1999-2008. https://doi.org/10.1109/infocom52122.2024.10621387

[35] Xu, K., Li, Y., Zhang, H., Lai, R. and Gu, L. (2022) EtinyNet: Extremely Tiny Network for TinyML. *Proceedings of the AAAI Conference on Artificial Intelligence*, **36**, 4628-4636. https://doi.org/10.1609/aaai.v36i4.20387

[36] Li, H., Guo, L., Yang, Y., *et al.* (2024) An Empirical Study of Rust-for-Linux: The Success, Dissatisfaction, and Compromise. 2024 *USENIX Annual Technical Conference* (*ATC*), California, 10-12 July 2024, 425-443.

[37] Hu, K., Wang, L., Mo, C. and Jiang, B. (2023) Work-in-Progress: Unishyper, a Reliable Rust-Based Unikernel for Embedded Scenarios. *Proceedings of the International Conference on Embedded Software*, Hamburg, 17-22 September 2023, 13-14. https://doi.org/10.1145/3607890.3608459

[38] Narayanan, V., Huang, T., Detweiler, D., *et al.* (2020) RedLeaf: Isolation and Communication in a Safe Operating System. 14*th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*), Virtual Event, 4-6 November 2020, 21-39.

[39] Ma, Z., Chen, G. and Zhong, L. (2023) Panic Recovery in Rust-Based Embedded Systems. *Proceedings of the* 12*th Workshop on Programming Languages and Operating Systems*, Koblenz, 23 October 2023, 66-73. https://doi.org/10.1145/3623759.3624549

[40] Sharma, A., Sharma, S., Tanksalkar, S.R., Torres-Arias, S. and Machiry, A. (2024) Rust for Embedded Systems: Current State and Open Problems. *Proceedings of the* 2024 *on ACM SIGSAC Conference on Computer and Communications Security*, Salt Lake, 14-18 October 2024, 2296-2310. https://doi.org/10.1145/3658644.3690275

[41] Ayers, H., Laufer, E., Mure, P., Park, J., Rodelo, E., Rossman, T., *et al.* (2022) Tighten Rust's Belt: Shrinking Embedded Rust Binaries. *Proceedings of the* 23*rd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, San Diego, 14 June 2022, 121-132. https://doi.org/10.1145/3519941.3535075