通过定向故障注入活动对 FreeRTOS 可信赖性的实验分析

Luca Mannella，Stefano Di Carlo，Alessandro Savino，都灵理工大学
控制与计算机工程系，意大利都灵{name.surname@polito.it}

arXiv:2603.25666v1 [cs.OS] 26 Mar 2026

摘要——**实时操作系统（RTOS）在安全关键领域发挥着至关重要的作用，在这些领域中，确定且可预测的任务执行至关重要。然而，它们日益受到电离辐射的影响，这可能损害系统可信性。为评估 FreeRTOS 在此类条件下的表现，我们提出 KRONOS：一种基于软件、非侵入式的传播后故障注入（FI）框架，可将瞬态和永久性故障注入操作系统（OS）可见的内核数据结构，而无需专用硬件或调试接口。我们使用 KRONOS 对 FreeRTOS 核心内核组件开展了大规模故障注入活动，包括调度器相关变量和任务控制块（TCB），以刻画内核级损坏对功能正确性、时序行为和可用性的影响。结果表明，指针和关键调度器相关变量遭到破坏时常会导致崩溃，而许多 TCB 字段对系统可用性的影响有限。**

索引词—**嵌入式系统、实时操作系统、故障注入、可靠性评估、单粒子翻转、恶劣环境**

I. 引言

实时嵌入式系统已成为现代关键基础设施的支柱，其应用范围远超消费电子领域，为汽车控制、航空航天与卫星系统以及其他平台提供支撑 [1]。这些系统广泛部署于安全关键和任务关键领域，因此必须对可信赖性作出严格保证，涵盖可靠性、安全性和容错性 [2], [3]。为满足这些应用多样且严苛的需求，嵌入式设备通常依赖专用软件栈。在某些情况下，软件以紧密耦合的应用专用固件形式实现；在另一些情况下，则构建于实时操作系统（RTOS）之上，由其统筹复杂的调度、资源管理和时序约束。

尽管设计不断进步，嵌入式系统仍容易受到内部故障（如硬件缺陷、软件漏洞和恶意篡改）以及运行环境带来的外部威胁影响 [2]。值得注意的是，瞬态故障和永久性故障，尤其是在太空和高空应用中，会产生重大影响；随着器件尺寸不断缩小，即使在地面，这类故障的发生率也在上升 [3]–[7]。在此背景下，辐射诱发效应，例如单粒子翻转（SEU）和持续性硬件故障，可能会传播至内核可见状态，并损害正确执行 [4]。现代嵌入式平台对此类现象日益脆弱，推动了稳健的可靠性评估与缓解技术的发展 [3]。

在现有方法中，故障注入（FI）被视为评估系统韧性的基石。通过有意引入故障（无论发生在硬件、软件还是系统层面），研究人员可以在受控条件下系统地研究故障传播、错误检测和恢复机制 [8], [9]。虽然许多故障注入研究着重于故障在应用层产生的影响，但也有一些研究强调，应考虑实时操作系统内核及其数据结构，例如调度器队列、定时器和任务控制块（TCB），因为这一层发生的故障会显著影响系统行为和安全性 [2], [9]。然而，对实时操作系统在瞬态和永久性故障下的表现及其对系统可信性的影响进行系统分析，仍是一个有待探索的研究方向。

本文研究影响实时操作系统关键内核数据结构的瞬态和永久性故障，在这些故障传播至实时操作系统可见状态后，如何影响系统可信性。本文的主要贡献包括：(1) 设计了面向 FreeRTOS 的内核级对象导向、非侵入式开放故障注入系统（KRONOS），这是一种新颖的基于软件的 FreeRTOS 故障注入框架 [10]；(2) 针对实时操作系统可见内核状态中的瞬态和永久性故障模型，对 FreeRTOS 开展了广泛的内核级故障注入活动；(3) 详细分析结果分布并识别关键内核数据结构，从而促进早期漏洞评估，并找出最敏感的 FreeRTOS 组件。

本文其余部分安排如下：第二节回顾故障注入（FI）相关研究。第三节详细介绍所提出的故障注入方法及其与 FreeRTOS 的集成。第四节描述实验设置，第五节展示并讨论故障注入活动的结果。最后，第六节总结全文。

II. 相关工作

故障注入是一种广泛使用的技术，通过有意向系统中引入故障，并观察系统在这些人为构造的故障场景下的行为，来评估系统的可信赖性 [11]–[15]。故障注入可按注入机制分为四类：(i) 物理故障注入、(ii) 基于硬件的故障注入、(iii) 基于软件的故障注入，以及 (iv) 基于模型的故障注入。物理故障注入涉及

本研究得到 COLTRANE-V 项目资助。该项目由大学与研究部在 PRIN 2022 计划（D.D.104 - 02/02/2022）框架内资助；本研究还在 Space It Up 项目框架内开展，该项目由意大利航天局 ASI 和大学与研究部 MUR 资助，合同编号为 2024-5-E.0，CUP 编号为 I53D24000060005。

让系统实现经受外部现象的影响，例如辐射。基于硬件的技术利用硬件接口（例如调试接口）修改寄存器内容或内存。基于软件的方法会改变软件层，以变量或内存缓冲区为目标。最后，基于模型的故障注入作用于抽象模型，例如硬件描述语言 (HDL) 或指令集模拟器，从而能够在仿真环境中开展故障注入活动。有关这些技术的更多详情，请参阅 [14], [15]。

针对操作系统层进行故障注入尤其具有挑战性 [16], [17]，，因为影响核心内核数据的故障可能导致不可恢复错误和系统挂起。虽然在应用层注入故障也可能产生类似影响，但这种情况出现的频率往往较低，而且处理挂起通常更加耗时，从而增加了故障注入活动的复杂性 [9]。

基于模型的故障注入框架近期取得的进展包括 gem5-MARVEL [18]。该框架基于 gem5 微体系结构模拟器 [19], ，可在模拟环境中针对多种指令集架构和系统组件（包括操作系统内核），在精细的硬件层级进行故障注入。尽管它支持精细的微体系结构故障建模和操作系统层故障注入，但这种方法需要较长的模拟时间，且复杂度较高。更广泛地说，微体系结构层的故障注入需要付出模拟时间较长的代价。

先前研究已广泛探讨硬件和微体系结构层面的故障注入，并经常指出简化的高层模拟方法存在局限。Papadimitriou 和 Gizopoulos [7] 在处理器微体系结构中进行了故障注入，并证明，纯软件层面的故障注入技术（通过翻转程序变量中的位来注入故障）可能忽略重要的微体系结构效应；与全栈分析相比，有时甚至会颠倒脆弱性排名。同样，Cho 等人 [20] 考察了架构层和内存层的故障注入方法，指出这类技术虽然执行速度快，但其错误率估计可能相差一个数量级。总体而言，这些研究凸显了高抽象层次的故障注入在捕捉真实硬件错误行为方面存在的固有局限，同时也未考虑操作系统层的故障处理机制。

相比之下，我们的工作以操作系统层为目标，直接向实时操作系统内核的内存结构注入故障，以评估系统软件本身的可信赖性。通过有意不进行微体系结构建模，所提出的方法专注于分析受损内部状态下的操作系统响应，例如故障条件下任务调度器、内核队列和内存管理器的行为。我们的目标并非估算硬件故障率，而是揭示硬件中心型故障注入无法观察到的操作系统级故障模式和鲁棒性局限。这一视角通过阐明故障如何在操作系统内部传播并显现，补充了现有的跨层故障分析。

然而，其他研究开始将目标转向操作系统结构。基于故障注入的自动化测试（FIAT） [21] 是最早针对内核的软件故障注入工具之一。它将抽象数据损坏模型应用于运行中的任务，但缺乏选择性破坏内部操作系统结构（如任务控制块）的精度。另一个基于微内核的框架是 MAFALDA [22]。尽管它可以向 API 参数或微内核内存段注入故障，但其架构将注入与执行分置于不同机器上，因此难以深入了解内部故障传播。RTOS Guardian [23] 为嵌入式实时操作系统引入了一种基于硬件的故障监测架构。其组件通过非侵入式监测系统总线来验证实时操作系统调度完整性。尽管该架构能够有效检测符合 IEC61000-4-29 [24], 标准的电磁感应故障，但其分析仅限于时间异常，无法扩展到驻留内存的内核结构损坏。

Mamone 等人 [25] 开展了最早一批通过故障注入分析 FreeRTOS 内核结构可靠性的实验研究之一。该研究表明，直接向实时操作系统的内部数据结构注入故障，对于识别关键组件具有重要意义；研究采用基于硬件的实验装置，并重点关注瞬态故障效应。我们的工作旨在从多个方面扩展这项分析，详情见第 III、IV 和 V 节。具体而言，KRONOS 无需专用硬件即可实现自动化、可重复的内核级故障注入，并支持瞬态和永久故障模型。此外，所提出的方法将实验范围扩展到更多种类的内核对象和执行条件，从而能够更全面地描述实时操作系统级故障模式。

III. 提出的方法

KRONOS 是一种故障传播后故障注入框架，作用于操作系统可见内存，并假设故障已到达内核可访问状态。这种抽象能够在不对底层传播过程建模的情况下，对内核数据损坏时实时操作系统的行为进行受控分析。它系统地揭示了最脆弱的内核结构和失效模式，从而对 FreeRTOS 组件进行有条件的脆弱性评估，而非评估绝对硬件故障率。

KRONOS 基于面向 Linux 和 Windows 的 FreeRTOS 移植版本构建，使我们能够在没有硬件的宿主操作系统上，将完整的 FreeRTOS 系统作为一组用户空间进程和线程运行。借助 FreeRTOS 可作为系统进程运行这一特性，KRONOS 完全基于软件实现，无需进行架构修改或使用外部调试接口。它具有良好的可移植性，且能够直接访问内核结构，因此非常适合对基于 FreeRTOS 的系统进行系统化的可靠性评估。具体而言，KRONOS 由以下部分组成：(i) 命令 模块，用于指定注入命令；(ii) 目标 模块，负责提取和获取注入目标；(iii) 一个操作系统抽象层，使注入器能够在不同的 FreeRTOS 移植版本上运行；(iv) 一个注入器模块；以及 (v) 一个日志模块，通过在关键事件（例如任务切换）处挂接到 FreeRTOS，记录运行时行为。图 1 展示了故障注入应用主要组件的概览。

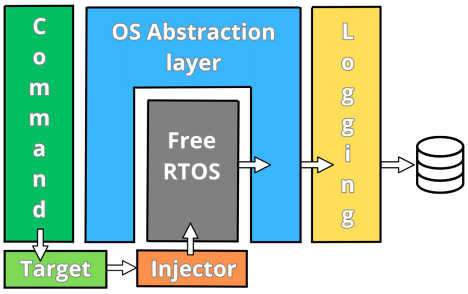

图 1. KRONOS 的主要组件及其交互概览。

A. 故障注入与结果收集工作流

在仿真端口中，FreeRTOS 以进程形式运行（且端口实现保证了节拍基准）。因此，KRONOS 被设计为在同一进程内的专用线程中运行。这种设计使注入器线程能够与仿真的 FreeRTOS 共享同一进程地址空间，同时不改变 FreeRTOS 的实现和配置。这种共享内存模型能够精确访问 FreeRTOS 关键组件中的故障注入，对系统的影响极小，且无需特殊硬件或架构修改。

故障注入工作流包含四个关键步骤：

(I) 注入目标指定：通过专用收集器函数，在运行时根据名称、基准内存地址、大小以及内核数据中的结构层次结构，识别符合条件的内核对象（变量、指针、列表、结构体）。

(II) 黄金运行分析：FreeRTOS 任务不间断地执行，以建立时序基线和预期输出，用于功能正确性比较。

(III)运行初始化：注入器加载目标参数（相对于调度器启动的注入时间偏移、字节/位偏移、故障类型）以及黄金运行配置文件，启动 FreeRTOS 调度器（允许任务/定时器正常执行），并休眠至指定的注入时间。

(IV) 故障注入：注入器唤醒并应用已配置的故障。

注入后，KRONOS 会监控调度器状态：如果调度器运行正常（只有空闲任务处于就绪状态），就让空闲任务钩子正常终止系统；否则，会在可配置的超时时间后强制关机。随后，根据黄金运行参考分析日志和输出，检查任务的正确性与执行时间，这与此前提出的更广泛的软件测试监控方法一致 [20]。

每次运行都可归入以下标准结果类别之一：

- **无害**：系统在预期时间内完成任务并产生正确结果。
- **延迟**：结果在功能上正确，但执行时间超过预先设定的截止时间（因此违反了性能或实时性要求）。
- **静默数据损坏（SDC）**：系统按时完成，但产生错误结果且未报告错误。为了更好地捕捉实时执行的细微差异，这类别有两个标签：如果静默数据损坏（SDC）伴随延迟产生，则相应标记为静默数据损坏延迟（SDC DELAY）。
- **挂起**：系统始终无法完成（没有输出或完成信号），通常是由于死锁、活锁或未处理故障所致。时间限制可配置为黄金运行时间的某个百分比。
- **崩溃**：系统突然终止（例如通过异常、中止或未处理信号），导致无法正常关闭并生成输出。
- **无效**：由于某些操作系统结构仅在特定用例行为下才是有效目标，因此，即使尝试对目标进行注入，只要该目标不被视为有效，我们就将其归入单独的类别。

B. 故障注入器实现

KRONOS 针对实时操作系统可见的内核状态实现了两种软件级故障模型：瞬态故障和永久性故障。这些模型并不模拟辐射诱发效应在电路级的物理产生过程；相反，它们描述的是故障已传播至内核数据结构并在架构层面上对实时操作系统可见的情形。从这个意义上说，所提出的故障注入方法是在传播之后进行的，目标是故障在软件中的表现。

瞬态故障 模拟内核目标发生的非持久性单粒子损坏，其表现为选定内核对象的运行时内存中发生一次位翻转。在配置的故障注入时刻，注入器直接访问内核进程内存空间，定位目标内存地址以及字节和位偏移量，并翻转目标位。注入之后，内核对同一位置的后续写入不会受到影响。该模型捕捉的是一种临时故障效应：它会体现在实时操作系统状态中，但不会被永久保留。

永久性故障用于模拟内核目标的持续性损坏，表现为目标内核对象中所选位的固定值故障。由于损坏后的值必须在后续更新中保持不变，KRONOS 引入了一种编译时补丁机制，在构建托管可执行文件之前预处理 FreeRTOS 内核代码。补丁程序使用 PCRE2 定位对目标位置的所有写入操作，并在每次写入后插入按位掩码，以强制施加固定值故障。这种策略确保在执行期间损坏位始终保持为固定值，代价仅为略微增加执行时间。因此，它模拟的是一种持续存在的软件可见故障状态，而不是最初导致该故障的底层物理机制。

IV. 实验设置

为了评估 FreeRTOS 的可信赖性，我们针对不同的内核结构开展了广泛的故障注入活动。具体而言，我们将分析目标分为四组：(i) 全局变量： 影响调度决策并有助于管理 FreeRTOS 的变量，如表 I 所述；(ii) 指针： 指向用于管理实时操作系统的 FreeRTOS 全局变量或数据结构的指针（见表 II）；(iii)列表：FreeRTOS 用于管理任务的结构。任务

表 I FreeRTOS 全局变量目标

目标	TYPE	描述
uxCurrentNumberOfTasks	无符号基础类型	活动任务的数量。
uxDeletedTasksWaitingCleanup	无符号基础类型	等待清理的任务控制块数量，由空闲任务执行。
xPendedTicks	节拍类型	累计节拍数 调度器处于阻塞状态时。
任务编号	无符号基础类型	任务创建计数器（唯一标识符）。
最高就绪优先级	无符号基础类型	至少有一个就绪任务的最高优先级级别。
下一个任务解除阻塞时间	节拍类型	下一个延迟任务到期解除阻塞时的节拍计数。
xTickCount	节拍类型	调度器启动以来的系统节拍计数器。
xNumOfOverflows	基本类型	xTickCount 已回绕的次数。
xSchedulerRunning	基本类型	调度器启动标志。
xTimerQueue	队列句柄	内部定时器命令的句柄队列。
xTimerTaskHandle	任务句柄	定时器守护任务句柄。
xYieldPending	基本类型	上下文切换待处理标志

表 II FreeRTOS 指针目标。

目标	描述
pxCurrentTCB	指向正在运行的任务控制块的指针任务。
pxCurrentTimerList	指向当前定时器列表的指针。
pxDelayedTaskList	指向延迟任务列表的指针。
pxOverflowDelayedTaskList	指向溢出延迟任务列表。
pxOverflowTimerList	指向当前溢出定时器列表的指针。
xIdleTaskHandle	指向空闲任务的任务控制块的指针

可以根据其状态属于不同的列表，即就绪、延迟、挂起或阻塞状态（见表 III）；(iv) 当前任务控制块：与当前运行任务相关联的数据结构（见表 IV）。其中包含有关任务的相关信息，即任务状态、优先级和栈指针。

实验设置采用 TACLeBench 套件中的五个基准测试，由优先级各不相同的任务执行 [26]：(i) SHA（经典哈希函数）、(ii) FFT（快速傅里叶变换计算）、(iii) CUBIC（三次方程求解器）、(iv) HUFF_DEC（霍夫曼解码）和 (v) ADPCM_ENC（自适应脉冲编码调制编码）。这些基准测试涵盖密码学、信号处理和

表 III FreeRTOS 列表目标。

目标	描述
pxReadyTasksLists	按任务优先级索引的就绪任务列表数组按任务优先级。
xDelayedTaskList1	列表 of 任务 延迟 via vTaskDelay()。
xDelayedTaskList2	列表 of 任务 延迟 via vTaskDelay()。
xPendingReadyList	变为就绪状态的任务，当时调度器已锁定。
xActiveTimerList1	活动软件定时器列表。
xActiveTimerList2	活动软件定时器列表。
xSuspendedTaskList	阻塞任务列表。
xTasksWaitingTermination	已删除任务挂起重新-由空闲任务清理已删除任务的资源。

表 IV FreeRTOS 当前任务控制块目标。可通过pxCurrentTCB访问。

目标	TYPE	描述
pcTaskName	*char**	当前任务名称。
pxStack	栈类型 *t**	指向当前任务栈底的指针。
任务标签	任务钩子函数 *t*	指向该任务应用程序任务标签的指针当前运行任务。
栈顶	栈类型 *t**	栈顶的指针的当前正在执行的任务。
ucDelayAborted	*uint8*	标记当前任务的阻塞延迟是否被提前中止。
ucNotifyState	*uint8**	任务通知机制使用的整数数组。
ulNotifiedValue	*uint32**	任务通知机制使用的整数数组。
运行时间计数器	*uint32*	累计任务处于运行状态的时间。
基准优先级	无符号基础类型	互斥锁优先级继承机制所使用的当前任务基准优先级互斥锁优先级继承机制。
持有的互斥锁数量	无符号基础类型	当前任务持有的互斥锁数量。
uxPriority	无符号基础类型	当前任务的优先级。
任务编号	无符号基础类型	用户定义的任务标签。
uxTCBNumber	无符号基础类型	内核分配的唯一 TCB ID。
xEventListItem	*ListItem*	事件（队列、信号量）的链表节点。
xStateListItem	*ListItem*	将 TCB 链接到调度器链表中的链表节点。

压缩工作负载，构成了多样化的任务组合。前三个任务（SHA、FFT 和 CUBIC）的优先级设为 1，HUFF_DEC 的优先级设为 2，ADPCM_ENC 的优先级设为 3。优先级经过有意区分，以便使用多个 就绪列表，并在故障注入过程中产生非平凡的抢占行为。每个任务执行一次计算后便自行删除，从而确保 FreeRTOS 调度器能够检测到任务完成并正常关闭。

FreeRTOS 内核配置为 1 kHz 的时钟节拍率、7 个优先级级别、协作式多任务处理、动态创建任务，并支持软件定时器、计数信号量、递归互斥锁和队列集。研究考虑了瞬态故障和永久性故障，故障注入发生在调度器启动后的 10,000 ns 窗口内（延迟偏差 = 5%，挂起时间为黄金运行时间的 = 300%）。每次运行都会执行五个 TACLeBench 任务以及系统/定时器守护任务，直至正常关闭。

为获得具有统计意义的结果，我们采用了来自 [27]（32 位参考）的二项比例估计方法，并将故障空间视为实际上无限。由此得出每个位置需进行 666 次故障注入（置信度为 99%，误差范围为 5%），所有目标共计进行 83,916 次故障注入。

为评估执行时间稳定性，我们使用四个并行工作进程执行了五次故障注入活动。平均实际运行时间为 76.7 s，标准差为 7.1 s（最小值–最大值：70.3–91.3 s）。这些结果证实，KRONOS 能够在切实可行的时间范围内执行大规模故障注入活动，同时确保不同独立注入样本之间具有可重复性，因此非常适合早期开发。

V. 实验结果

实验活动表明，在操作系统可见的内核内存损坏情况下，FreeRTOS表现出中等程度的容错性，但多个内核组件仍然极易受到影响（表V）。约70%的运行成功完成，约3%出现执行延迟，

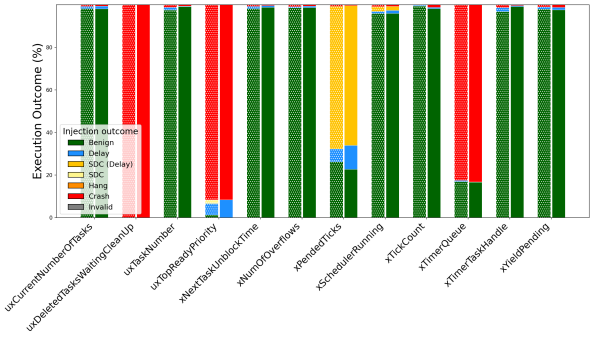

图2. 对*FreeRTOS* **变量**的注入：瞬态故障（左侧）和永久故障（右侧）。

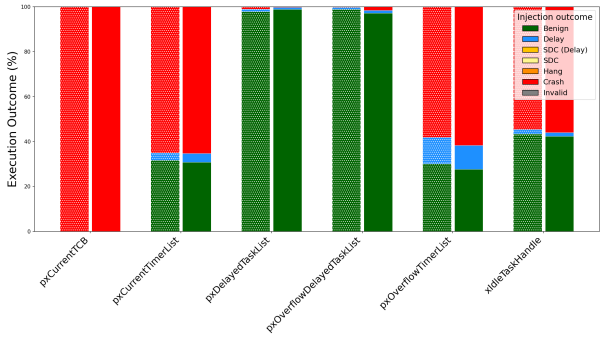

图3. 对*FreeRTOS* **指针**进行的注入：瞬态（左侧）和永久（右侧）故障。

超过20%导致系统崩溃，表明默认配置下的故障处理能力有限。静默数据损坏（SDC）事件很少发生（低于2%），且通常与时序偏差有关；不到5%的故障注入针对无效对象。大多数内核组件在瞬态故障和永久性故障下的故障率相近，因为关键数据结构损坏常常会立即导致不可恢复的故障，例如调度器不一致或无效内存访问。因此，在一次执行过程中，故障持久性通常不会实质性地改变观察到的运行级结果。这些发现表明，实时操作系统可靠性高度依赖于工作负载特征和运行环境。

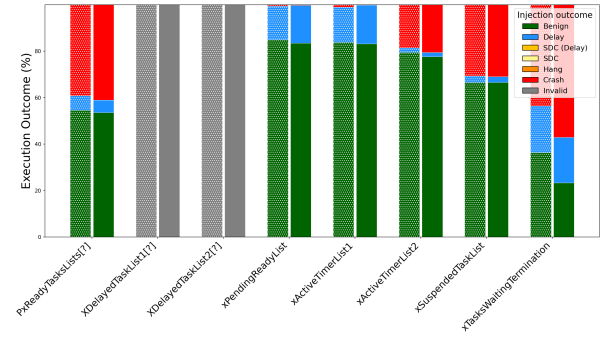

图4. 对*FreeRTOS* **列表**进行的注入：瞬态（左侧）和永久（右侧）故障。

表V 按故障类型划分的整个故障注入活动结果。

故障类型	无害	延迟	SDC	静默数据损坏（延迟）	HANG	崩溃/无效	
瞬态	70.16%	2.80%	0.04%	1.69%	0.00%	20.43%	4.88%
永久	69.66%	3.00%	0.00%	1.65%	0.00%	20.82%	4.88%

基于类别的分析表明，若干变量（如 uxDeletedTasksWaitingCleanup、最高就绪优先级和 xTimerQueue）一旦被修改，几乎都会导致故障，且几乎总是引发系统崩溃。这凸显了它们在调度器核心决策和资源处理决策中的关键作用。对不太敏感的变量（如 xTickCount 或 uxCurrentNumberOfTasks）进行修改，通常仍能正确执行，但在特定时序条件下偶尔会导致崩溃或延迟。值得注意的是，修改 xPendedTicks 会造成很大比例的静默数据损坏，且总是伴随延迟（见图2）。

正如预期，指针 变量是对故障最敏感的元素：位翻转几乎总会导致系统崩溃，因为它们会立即使对象或列表引用失效。指向动态调度器、延迟列表或定时器列表的指针最为敏感，而 pxCurrentTCB 则最为关键，其损坏会导致两种故障类型下的所有运行均发生崩溃。其他敏感元素包括 pxCurrentTimerList 和 xIdleTaskHandle，它们被修改时，会导致超过 50% 的系统崩溃。值得注意的是，影响这个变量类别不会导致任何静默数据损坏（图3）。

与另一类别相比，注入列表对象的故障较少导致崩溃（约为 20% 至 50%），但有一小部分会造成延迟（见图4）。然而，该类别中的崩溃率往往会因永久性故障而上升。向任务等待终止列表（即 xTasksWaitingTermination）注入故障，表现出极高的敏感性。相比之下，与不太活跃的内核操作相关的列表（例如 xSuspendedTaskList）表现出较高的正确执行率，尽管系统崩溃的风险仍不可忽视。值得注意的是，即使影响此类变量，也不会造成任何静默数据损坏。

如图4所示，在注入时有些目标从未处于有效状态。具体而言，延迟任务列表为空，因为所采用的用例主要以计算为主，在选定的注入时刻无需任务进入延迟状态。

图5显示，破坏当前任务控制块中的字段并未产生特别敏感的影响；在大多数情况下，崩溃和延迟的比例都很低。最敏感的字段是 uxPriority，修改该字段会导致超过 80% 的运行发生崩溃。这一证据凸显了优先级管理对于确定性实时操作系统运行的重要性。此外，修改 ucDelayAborted 会导致超过 40% 的运行发生崩溃。值得注意的是，即使影响任务控制块中的字段，也不会造成静默数据损坏。

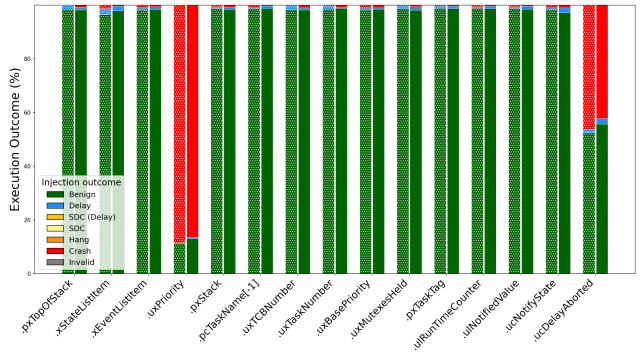

图 5. 对 *FreeRTOS* **当前任务控制块** 字段进行注入：瞬态故障（左侧）和永久性故障（右侧）。

VI. 结论

本研究使用 KRONOS 扩展了此前对 FreeR-TOS 可靠性的实验分析。KRONOS 是一种系统化的基于软件的故障注入框架，面向内核级数据结构。通过操作系统可见的内存状态，KRONOS 无需专用硬件或调试接口，即可对实时操作系统故障模式进行可重复评估。

针对关键 FreeRTOS 内核对象开展的大规模故障注入活动表明，尽管许多损坏都能被容忍，但核心调度器相关结构仍然十分脆弱，常常导致崩溃或截止期限违例。瞬态故障与永久性故障造成的影响相似，这表明关键实时操作系统状态遭到破坏时，往往会立即引发不可恢复故障。分析结果凸显了当故障变得体系结构可见时 FreeRTOS 内核组件所呈现的条件性脆弱性，并补充了较低层次的故障传播研究，有助于实时操作系统设计人员识别在安全关键和任务关键系统中需要加强保护或监控的内核结构。

未来工作包括将 KRONOS 扩展至其他实时操作系统，并集成跨层模型，以关联操作系统可见故障与特定设备级现象。

致谢

作者感谢 Giovanni De Florio 和 Dimitri Schiavone 在攻读硕士学位期间为本研究作出的贡献。

参考文献

[1] C. Chen 等人，“DeepDriving：学习自动驾驶中的直接感知可供性”，载于*2015*年*IEEE*国际计算机视觉会议（ICCV），2015年12月，第2722–2730页。 [2] M. A. Solouki 等人，“嵌入式系统中的可信赖性：容错方法与基于软件的缓解技术综述”，*IEEE Access*，第12卷，第180 939–180 967页，2024年。 [3] R. Aalund 和 V. Philip Paglioni，“提升嵌入式系统硬件可靠性：文献综述”，*IEEE Access*，第13卷，第17 285–17 302页，2025年。 [4] J. R. Srour 和 J. W. Palko，“辐照半导体器件中的位移损伤效应”，*IEEE*核科学汇刊，第60卷，第3期，第1740–1766页，2013年6月。 [5] A. Vallero 等人，“SyRA：针对微处理器系统存储器阵列跨层软错误弹性的早期系统可靠性分析”，*IEEE*计算机汇刊，第68卷，第5期，第765–783页，2019年5月。

[6]M. Portolan 等人，“用于早期安全性评估的故障注入替代方案”，载于2019年IEEE欧洲测试研讨会（ETS），2019年，第1–10页。 [7] G. Papadimitriou 和 D. Gizopoulos，“揭开系统脆弱性栈的面纱：各层的瞬态故障效应”，载于2021年ACM/IEEE第48届年度国际计算机体系结构研讨会（ISCA），2021年，第902–915页。 [8] A. Benso 和 S. Di Carlo，“故障注入艺术”，《控制工程与应用信息学》，第13卷，第4期，第9–18页，2011年。 [9] C. De Sio 等人，“评估嵌入式系统对单粒子效应的可靠性：实时操作系统与裸机方案的比较”，《微电子可靠性》，第150卷，文章编号115124，2023年，欧洲电子器件可靠性、失效物理与分析第34届研讨会（ESREF 2023）特刊。 [10] F. Guan 等人，“作为实时操作系统演进案例研究的开源 FreeRTOS”，《系统与软件杂志》，第118卷，第19–35页，2016年。 [11] G. A. Kanawati 等人，“FERRARI：灵活的基于软件的故障与错误注入系统”，《IEEE计算机汇刊》，第44卷，第2期，第248–260页，1995年2月。 [12] J. Carreira 等人，“Xception：一种评估现代计算机可信赖性的实验技术”，《IEEE软件工程汇刊》，第24卷，第2期，第125–136页，1998年2月。 [13] M. Ebrahimi 等人，“快速、灵活且易于开发的基于FPGA的故障注入技术”，《微电子可靠性》，第54卷，第5期，第1000–1008页，2014年。 [14] G. Di Natale 等人，《计算系统的跨层可靠性》。工程技术学会（IET），2020年。 [15] P. Bodmann 等人，“软错误对Arm微处理器的影响：早期估算与芯片测量”，《IEEE计算机汇刊》，第1–1页，2021年。 [16] A. Bosio 等人，“评估嵌入式系统中的FreeRTOS可靠性”，载于IEEE/IFIP第52届年度国际可信系统与网络会议增刊（DSN-S），2022年，第28–30页。 [17] E. Casseau 等人，“特别专题：受测操作系统概览：操作系统对计算连续体韧性的重要性”，载于2021年IEEE第39届超大规模集成电路测试研讨会（VTS），2021年，第1–10页。 [18] O. Chatzopoulos 等人，“Gem5-MARVEL：异构片上系统架构的微架构级韧性分析”，载于2024年IEEE国际高性能计算机体系结构研讨会（HPCA），2024年3月，第543–559页。 [19] N. Binkert 等人，“gem5模拟器”，《SIGARCH计算机体系结构新闻》，第39卷，第2期，第1–7页，2011年8月。 [20] H. Cho 等人，“鲁棒系统设计中软错误注入技术的定量评估”，载于第50届年度设计自动化会议论文集，丛书DAC ’13。美国纽约：计算机协会（ACM），2013年。 [21] J. Barton 等人，“使用FIAT进行的故障注入实验”，《IEEE计算机汇刊》，第39卷，第4期，第575–582页，1990年4月。 [22] J. Arlat 等人，“商用现成微内核系统的可信赖性”，《IEEE计算机汇刊》，第51卷，第2期，第138–163页，2002年2月。 [23] D. Silva 等人，“一种用于基于实时操作系统的嵌入式系统的故障检测硬件方法”，载于2011年第16届IEEE欧洲测试研讨会。IEEE，2011年5月，第209–209页。 [24] 国际电工委员会（IEC），“IEC 61000-4-29：电磁兼容性（EMC）—第4-29部分：试验和测量技术—直流输入电源端口抗扰度试验中的电压暂降、短时中断和电压变化”，2000年。 [25] D. Mamone 等人，“嵌入式系统中实时操作系统可靠性的分析”，载于2020年IEEE超大规模集成电路与纳米技术系统缺陷与容错国际研讨会（DFT）。IEEE，2020年，第1–6页。 [26] H. Falk 等人，“TACLeBench：支持最坏情况执行时间研究的基准测试集合”，载于第16届最坏情况执行时间分析国际研讨会（WCET 2016），信息学开放获取丛书（OASIcs），M. Schoeberl 编，第55卷。德国达格施图尔：施洛斯·达格施图尔—莱布尼茨信息学中心，2016年，第2:1–2:10页。 [27] R. Leveugle 等人，“统计故障注入：量化误差与置信度”，载于2009年欧洲设计、自动化与测试会议暨展览。IEEE，2009年4月，第502–506页。