哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

Del0n1x

现场赛文档

参赛队名 Del0n1x

队伍成员 姚俊杰、卢家鼎、林顺喆

指导老师 夏文、仇洁婷

8.20 现场赛文档

一、现场赛成绩

在 8.20 内核赛道现场赛中，Del0n1x 总分为 145，排名第 8。排名与测例通过情况如下图：

比赛提交到排行榜更新有20秒左右的延迟

#	参赛ID	参赛用名	QEMU-Riscv	QEMU-LoongArch	开发板-Riscv	开发板-LoongArch	得分(DESC)
1	T202510003996120	Starry Mix/ 清华大学	100.00	100.00	70.00	0.00	270.0000
2	T202510336995214	NoAxiom/ 杭州电子科技大学	60.00	60.00	60.00	60.00	240.0000
3	T202510003995291	undefined/ 清华大学	90.00	90.00	60.00	0.00	240.0000
4	T202510213995926	火箭队/ 哈尔滨工业大学	80.00	80.00	55.00	0.00	215.0000
5	T202518123995568	Chronix/ 哈尔滨工业大学（深圳）	60.00	60.00	40.00	50.00	210.0000
6	T202518123995755	rustflyer/ 哈尔滨工业大学（深圳）	60.00	60.00	60.00	10.00	190.0000
7	T202510336995486	StarryX/ 杭州电子科技大学	50.00	50.00	45.00	0.00	145.0000
8	T202518123995600	Del0n1x/ 哈尔滨工业大学（深圳）	50.00	45.00	50.00	0.00	145.0000

Del0n1x 现场赛排行榜情况

平台	测例	完成情况
Qemu-rv	git	✅ Task0 ✅ Task1 ❌ Task2
	vim	✅ Task0 ❌ Task1
	gcc	✅ Task0 ✅ Task1
	rustc	✅ Task0 ❌ Task1
Qemu-la	git	✅ Task0 ✅ Task1 ❌ Task2
	vim	✅ Task0 ❌ Task1
	gcc	✅ Task0 ✅ Task1
	rustc	❌ Task0 ❌ Task1
开发板-rv	git	✅ Task0 ✅ Task1 ❌ Task2
	vim	✅ Task0 ❌ Task1
	gcc	✅ Task0 ✅ Task1
	rustc	✅ Task0 ❌ Task1
开发板-la		测例运行失败

Del0n1x 现场赛通过情况

二、各平台适配历程

RISC-V Qemu virt

RISC-V Qemu 是我们最熟悉的平台。在 Qemu 平台上我们实现了设备树解析和 RISC-V PLIC 中断控制器驱动，支持外设中断，适配了 virtio 相关驱动和串口驱动，并基于 Qemu 平台测试了绝大部分的初赛及决赛预发布测例，验证了作品的稳定性。在现场赛中，我们大部分的开发过程都基于 Qemu 事先测试。

LoongArch64 Qemu virt

LoongArch Qemu 的情况有所不同。LoongArch Qemu 采用和真实开发板截然不同的外设平台（设备树）和中断控制模型，部分设备（如 PCH-PIC 中断控制器）的实现细节与真实设备的用户手册无法对应。同时，Qemu 上没法很好地支持地址不对齐读写问题的测试，这虽然对我们前期的开发提供了便利，但为我们现场赛阶段的失利埋下了伏笔。

我们通过参考龙芯技术手册、阅读设备树文件、阅读 Qemu 源代码、参考往届作品等方法，在初赛阶段熟悉了 LoongArch Qemu virt 平台的技术细节，实现了 virtio PCI 驱动、virtio 块设备驱动、ns16550 串口驱动的支持，设计并编写了支持 UART 中断的中断控制器驱动集，在现场赛初期获得了不错的成绩。

RISC-V 星光二代板

星光二代板的外设平台和 RISC-V Qemu 基本相似，我们的设备子系统只需稍加配置就能直接启动。现场赛前，我们初步完成了星光二代板串口和 SD 卡块设备驱动的适配和调试。在现场赛中，我们可以直接烧录测例文件系统镜像到 SD 卡中，由于平台差异较小，我们事先在 Qemu 虚拟平台上保证测例通过，在上板测试过程并未遇到很大的问题。

龙芯 2K1000 开发板

我们在赛前对龙芯 2k1000 开发板的适配投入了大量的时间，实现了中断控制器驱动，封装和适配了赛方提供的 AHCI 块设备驱动，支持了地址不对齐读写例外的处理，重写了设备子系统以支持跨平台启动系统。但是在现场赛适配测例的过程中，我们的作品在 2k1000 开发板上遇到了无法预测的内存问题。该问题导致我们的操作系统在运行用户态程序时出现诡异 page fault，但通过反汇编定位错误位置、使用 trap context 和 gdb 阅读用户例外前寄存器后，却发现该位置只访问了合法地址。经过数小时的挑战，最终我们放弃了对 2k1000 开发板的适配，证明了我们赛前准备的严重不足。

三、测例适配历程

1. vim

在前期的准备中，我们完成了各平台的外设中断控制器驱动，成功在 Qemu 和开发板上实现了外部中断的使能和处理。我们开发了功能完备的 tty 子系统，实现了中断驱动的异步 IO 机制，通过对 ppoll、pselect、ioctl 等系统调用的适配，成功运行了功能完整的 busybox vi，这也使得我们在现场赛中可以很顺利地启动 vim、使用 vim 编辑、保存 vim 编辑的内容、使用 vim 的快捷键和各种模式。但是，在退出 vim 的过程中，我们遇到了无法预测的问题（疑似调度问题），所以很遗憾并未得到所有的分数。

2. gcc 与 rustc

gcc 与 rustc 的正确实现与文件系统相关。在初赛和决赛第一阶段，我们的文件系统功能基本完善。但在现场赛阶段，由于我们在赛前没有适配符号链接，导

致赛中我们发现编译器无法直接访问共享库。我们在现场赛采用在内核中硬编码转发的方式进行适配，成功让 gcc 编译了可运行的 hello world 程序，算是不幸中的万幸。

3. git

git 程序本地仓库的构建与使用与文件系统的正确实现。在 git 中使用大量的 rename 系统调用作为原语，借助操作系统的功能实现对文件的安全更新。我们在赛前便实现了本地运行 git 相关的系统调用，并且在 git 适配的过程中，我们通过阅读 git 源代码与其他调试手段，修复了大量实现有错的系统调用。遗憾的是，由于时间不足，我们放弃了联网的 git 操作的实现。