

机顶盒软件的坑

- 吉光片羽

2015年
hubugui@gmail.com
<https://goo.gl/Kz3lbv>

背景

- 时间

- 2004 ~ 2006

- 行业

- 广电业数字机顶盒

- 技术方案/DVB-C

- STi5518-标清

- 80MHZ/32MB/MPEG2

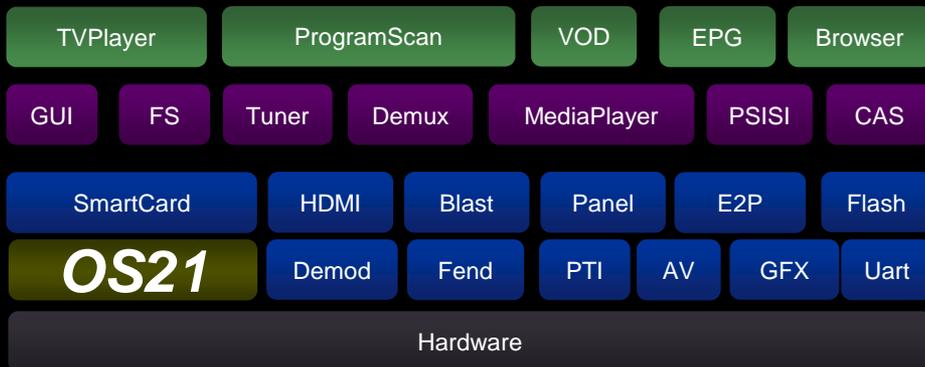
- STi7101-高清

266MHZ/128MB/H264

- 创业公司

- 3个软件、1个硬件

- 年轻、积累少、空间大、自驱动、疯狂



特点

- 嵌入式环境
 - OS21操作系统，任务实时优先级
 - 整体合计50万行代码，200以上线程
- 应用软件位于内核空间
 - 任何异常直接导致系统崩溃
 - 任何不良操作扩散后难定位
- 移植第三方闭源软件
 - 跨平台，挑剔移植层
 - 考验系统架构和多任务设计

系统崩溃

- 珍贵的现场
 - 寄存器
 - 任务栈和状态
 - 内存
- 符号表
- 日志

“只要敢重现，咱就能收拾”

现场被破坏

- 可重现
 - 解决速度依赖于重现周期
- 不可重现
 - 这是功能
- 难重现
 - 善用工具
 - 加入调试信息
 - 评审最近提交，逐步缩小范围
 - 换板子，放一放，等机灵



挑战

- 现场被破坏
- 加入调试信息
 - 崩溃位置和重现规律却发生变化
- 代码分析困难
 - 勇敢重构，核心模块结对
- 进度压力大
- 梦中也有她，失眠



“困难是拿来调戏的”

“只要可重现，咱就能搞掂”



“夜深了，睡觉才有好脑子，明天没准只要一刻钟”

Coders At Work



Seibel

"你曾跟踪调试过的最糟糕的bug是什么？"
"你是如何跟踪调试这个问题的？"



Thompson

"内存破坏，由硬件故障引起的，而非野指针"
"直到我写密集的乘除法程序，才让bug再现频率提高。以前是每隔几天崩溃，现在每隔几分钟，只有再现她才有继续排查的机会"

内存

- 没释放
 - 记录MMT(分配释放表), 一目了然
- 内存被破坏
 - 内存块头部放入魔数, 校验嫌疑模块
 - 校验栈溢出
- 堆破坏或释放错误地址
 - 地址校验
 - 小型堆校验, 尽早赶到破坏现场
 - 使用MMT定位脏节点的分配者
- 碎片
 - 数组/预分配

锁

- 保护资源
 - 一把锁保护多个资源：简单、粗粒度
 - 多把锁保护多个关系不清晰的资源，容易死锁
- 死锁
 - 修改锁的源码，记录占用任务ID
- 优先级翻转
 - 优先级继承：`mutex_create_priority`
 - 关键函数手动提升优先级或`task_lock`

任务

- 目的明确
- 优先级
 - 谨防“饥饿”
- 栈大小
- 生命周期
 - 至少有让出CPU的条件
- 通信手段
 - 消息队列不仅触发信号，还可以携带现场数据

优化

- 运行
 - *Profiler*工具找热点
 - 临时高优先级(CPU密集型)
- 设计
 - 算法改善
 - 多路IO复用
 - 同步?异步?
 - 减少任务切换
 - 性能评估机制
- 先做对, 再做好

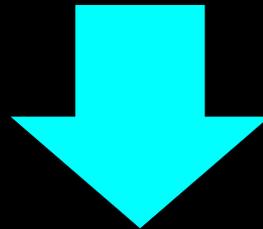
例1-符号重定向

- 现象
 - *strcpy*函数崩溃，栈被破坏
- 原因
 - 非法参数
- 对策
 - 符号重定向到*foobar_strcpy*，加入*trace*

例2-抱残守缺-为故障而设计



- 移植浏览器时要求PTI芯片(类似网卡)过滤DSMCC数据，但PTI不定期向主CPU传送错误数据，导致打不开网页
- PTI芯片源码不公开，创业公司没有方案厂商的给力支持



临时对策(基友Zeloz贡献)

- **错误发生时重置PTI驱动，可保障60%的情形正常**



终极方案：6个月后ST公司发布新版本

墨菲定律

- 担心的事总会发生
- 担心意味着存疑
- 经验丰富的你为什么存疑？
 - 自觉琢磨不够，存在未知和漏洞
 - 潜意识判断自有逻辑，故障没发生是时机未到

没有银弹

- 怀疑自己甚于他人
- 单元测试-吃自己的狗粮
- 闭源软件及周边容易出故障
- 设计评审，关键场景结对
 - 屈侠：“咨询师的介入姿势”
- 重构
 - 大刘：“无法维护不如尽早推倒重来”
- 良好的设计与实现远胜事后调试
- 慢就是快-慢点提高质量，整个进度反而快