第一届
开源操作系统
设计与分析
学术会议论文集

史元春　李明树　陶　品　编

The First Open Source
Operating System
Design and Analysis
Conference

OSDA
2013

清华大学出版社

第一届
开源操作系统
设计与分析
学术会议论文集

史元春　李明树　陶　品　编

清华大学出版社
北京

内 容 简 介

操作系统的设计与分析是计算机技术的核心技术之一，操作系统的安全可控性是事关国家安全的战略问题。当前计算机和各种电子设备已经深入应用到了国民经济的各行各业，同时也与社会大众的日常工作、学习、生活密切相关，这些设备中几乎都运行着各种不同类型和不同版本的开源操作系统，安全问题更是日益凸显。第一届开源操作系统设计与分析学术会议（The First Open Source Operating System Design and Analysis Conference，OSDA 2013）首次凝聚国内研究力量面向国家战略需求研讨学术问题，本会议论文集收录了 OSDA 2013 的 25 篇论文，内容涵盖了当前被广泛应用的 Linux 和 Android 两种开源操作系统，涉及开源操作系统的分析、安全、优化和增强等问题，对于推动开源操作系统技术的研究与学术交流具有重要的意义，本论文集反映了我国在上述领域的研究现状和最新进展。

图书在版编目（CIP）数据

第一届开源操作系统设计与分析学术会议论文集/史元春，李明树，陶品编. —北京：清华大学出版社，2014

ISBN 978-7-302-35435-2

Ⅰ. ①第… Ⅱ. ①史… ②李… ③陶… Ⅲ. ①操作系统–学术会议–文集 Ⅳ. ①TP316-53

中国版本图书馆 CIP 数据核字（2014）第 023052 号

责任编辑：白立军　顾　冰
封面设计：傅瑞学
责任校对：焦丽丽
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：河北新华第一印刷有限责任公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：11.75　　字　　数：280 千字
版　　次：2014 年 5 月第 1 版　　印　　次：2014 年 5 月第 1 次印刷
印　　数：1～200
定　　价：49.00 元

产品编号：057047-01

会议组织情况

大会主办单位

清华大学
中国科学院软件研究所

大会承办单位

清华大学

第一届开源操作系统设计与分析学术会议

大会主席

史元春　　清华大学
李明树　　中国科学院软件研究所

大会秘书长

贺也平　　中国科学院软件研究所

程序委员会

委　员：贺也平　　中国科学院软件研究所
徐明伟　　清华大学
陈文光　　清华大学
孙立峰　　清华大学
武延军　　中国科学院软件研究所
吴敬征　　中国科学院软件研究所
于佳耕　　中国科学院软件研究所
赵　珊　　中国科学院软件研究所
周启明　　中国科学院软件研究所
杨维康　　清华大学
向　勇　　清华大学
陶　品　　清华大学
白晓颖　　清华大学
王瑀屏　　清华大学
周悦芝　　清华大学
刘卫东　　清华大学
吕勇强　　清华大学
董　渊　　清华大学
陈　渝　　清华大学
诸葛建伟　清华大学
陈　康　　清华大学
宋佳兴　　清华大学
王道顺　　清华大学
王生原　　清华大学

前　言

第一届开源操作系统设计与分析学术会议（The First Open Source Operating System Design and Analysis Conference，OSDA 2013）于 2013 年 9 月 14 日在北京举办。当前计算机和各种电子设备已经深入应用到了国民经济的各行各业，同时也与社会大众的日常工作、学习、生活密切相关，这些设备中都运行着必不可少的操作系统软件，因此操作系统的安全可控性是国家关注的重要战略问题。

第一届开源操作系统设计与分析学术会议以 Linux 和 Android 两种开源操作系统为主要研究对象，与会学者共同探讨了开源操作系统分析和安全性评估的新理论、新方法和新技术。会议邀请了中国信息安全测评中心张普含副研究员和中国科学院信息工程研究所龚晓锐分别做了题为“软件源代码安全性分析”和“Android 应用安全分析——方法与挑战”的大会特邀报告，大会还围绕 Linux 和 Android 系统的分析、安全、优化和增强等问题，分为四个分会场展开了热烈的研讨。来自清华大学、中国科学院软件研究所、北京航空航天大学、北京理工大学、西安电子科技大学、信息产业部软件与集成电路促进中心等多所国内高校和科研单位的专家学者和研究生 50 余人参加了本次会议。

此次会议在筹备过程中得到了清华大学、中国科学院软件研究所和相关单位及各界人士的鼎力支持，在此表示衷心的感谢！此次会议的顺利召开，离不开 OSDA 2013 审稿专家对论文的认真评审，以及 OSDA 2013 会务工作组的辛勤工作，在此对陶品、吕勇强、白晓颖、陈渝、孙金龙、宋克清、刘馨等教师和学生志愿者的默默奉献表示由衷的敬意！

大会主席：史元春　贺也平

目　录

Android 平台常见文件系统性能分析与对比 ..... 原仓周　刘雨骁　张建　王雷（1）
Linux 系统调用引发的内存错误检测 ..... 徐永健　陈渝　王丹　肖奇学（8）
Linux 内核中整数溢出漏洞的检测研究 ..... 范文良　肖奇学　王欢　陈渝（14）
真实系统中的全自动内核动态跟踪分析 ..... 吴竞邦　向勇　陆慧梅　李玲（20）
Bash 软件包中的安全漏洞分析 ..... 光大昕　范建明　陶品（26）
init 进程执行流程分析 ..... 陶品　李明（31）
Linux 内核编译 make 参数优化 ..... 吴振亚　孙菁　李航（37）
一种针对内核模块的高代码覆盖率的动态检测方法
..... 李红鞠　吴敬征　罗天悦　曲璐（43）
一种 Linux 启动综合优化技术的研究 ..... 吴永强　李鹏飞　刘中山　苏锐丹（53）
基于云环境的 Android 应用程序安全隐患检测方法
..... 武志飞　曲璐　杨牧天　武延军（61）
Libkmod 的结构及安全漏洞分析 ..... 宋思超　陶品　李明（68）
GRUB 类 Bash 环境的安全性问题分析 ..... 冯立新　谢文学　陶品　肖奇学（73）
一种 Android 应用程序 DOS 漏洞和信息泄露漏洞的检测方法
..... 武志飞　吴敬征　罗天悦　杨牧天（78）
基于 Smack 模型的 Android 系统安全性增强方法
..... 李红鞠　杨牧天　武志飞　武延军（85）
针对 SQLite 弱权限导致隐私泄露威胁的检测方法
..... 曲璐　吴敬征　杨牧天　罗天悦（92）
Dalvik JNI 模块分析 ..... 褥震　王鹏　杨述　董渊　王生原（101）
Android 智能手机的电源管理和功耗分析
..... 苏静芳　吕晓娟　李兴华　杨维康　陈康　吕勇强（108）
基于同源性的 Android 操作系统源代码静态检测结果的分析
..... 罗天悦　吴敬征　李红鞠　武延军（115）
基于使用率和尾功耗的智能手机功耗监测模型
..... 杜园园　黄明义　吕勇强　王丹　陈渝（125）
Android 进程间通信性能分析 ..... 原仓周　冯力　郭飞　梁栋　王雷（132）
智能终端网络功耗优化技术研究 ..... 杜园园　黄明义　吕勇强　王丹　陈渝（139）
针对 Android 应用程序冗余授权的安全机制研究 ..... 金涛　高超　刘卫东（146）
智能手机尾功耗评估模型
..... 李建州　李兴华　杜园园　苏静芳　杨维康　吕勇强（153）
Android 知识产权风险分析与对策 ..... 蔡智　范兵　谢学军（160）
Dalvik 与 Java 虚拟机性能对比分析 ..... 骆欢　蒋挺宇　李叠　董渊　王生原（166）
附录 A　第一届开源操作系统设计与分析学术会议部分成员合影 ..... （176）

Android 平台常见文件系统性能分析与对比

原仓周[1] 刘雨骁[1] 张 建[2] 王 雷[2*]

[1]北京航空航天大学 软件学院，北京，100191

[2]北京航空航天大学 计算机学院，北京，100191

*通信作者，wanglei@buaa.edu.cn

摘 要：Yaffs 是第一个专门为 NAND Flash 存储器设计的嵌入式文件系统，适用于大容量的存储设备，在 Android 系统中得到了广泛的应用。本文首先分析了 Yaffs 文件系统的工作原理和性能特点，测试了 Yaffs 在不同平台环境下的读写性能。然后对比其他文件系统，对 Yaffs 进行了一个全面的性能测试。通过 Yaffs 和 Ext2、Ext3、Ext4 的读写性能对比实验，发现 Yaffs 文件系统的写入速率明显快于其他三种 Ext 文件系统，但在读取速率上却不及三种 Ext 文件系统。在函数级别的测试中，Yaffs 的速率要比其他三种文件系统低很多。后来对比 Yaffs 和 Ext4、NTFS、FAT32 文件系统在读写小文件小数据和读写大文件大数据时的性能差异，发现 Yaffs 和 NTFS 在文件大小不同时读写速率比较稳定，而 Ext4 和 Fat32 在小数据小文件时读写速率较低，随着数据量的上升，速率随之增大。

关键词：文件系统；Yaffs；性能分析；Android

1 引言

Linux 支持各种各样的具体文件系统，FAT、Ext2、Ext3、Ext4。Yaffs 是其中一种基于 Android 平台的具体文件系统。Yaffs（Yet Another Flash File System）是第一个专门为 NAND Flash 存储器设计的嵌入式文件系统，适用于大容量的存储设备；并且是在 GPL（General Public License）协议下发布的，可在其网站免费获得源代码。

Yaffs 是基于日志的文件系统，提供磨损平衡和掉电恢复的健壮性。它还为大容量的 Flash 芯片做了很好的调整，针对启动时间和 RAM 的使用做了优化。它适用于大容量的存储设备，已经在 Linux 和 WinCE 商业产品中使用。

我们首先通过 Yaffs 文件系统源码，分析其工作机制和工作原理。然后测试了 Yaffs 在不同平台环境下的读写性能，分析其实验结果。之后我们又通过 Yaffs 和 Ext2、Ext3、Ext4 的读写性能对比以及 Yaffs 和 Ext4、NTFS、FAT32 文件系统在读写小文件小数据和读写大文件大数据时的性能差异两个对比实验，更全面地分析了 Yaffs 文件系统的性能以及在不同条件下相对于其他文件系统的性能优劣。

论文接下来按照以下形式组织：第 2 节对其他的文件系统做简要介绍，为实验提供理论基础；第 3 节从文件操作、物理页的管理、垃圾回收以及安全性四个方面详细的分析 Yaffs 文件系统的工作机制；第 4 节描述了实验方案的设计与实现以及实验结果的分析；第 5 节总结全文并对未来工作的展望。

2 其他文件系统概述

2.1 FAT32

FAT32 是 Windows 系统硬盘分区格式的一种。这种格式采用 32 位的文件分配表，使其对磁盘的管理能力大大增强，突破了 FAT16 对每一个分区的容量只有 2GB 的限制。

2.2 NTFS

NTFS 是 Windows NT 以及之后的 Windows 的标准文件系统。NTFS 取代了 FAT 文件系统，为 Microsoft 的 Windows 系列操作系统提供文件系统。NTFS 对 FAT 和 HPFS（高性能文件系统）作了若干改进，例如，支持元数据，并且使用了高级数据结构，以便于改善性能、可靠性和磁盘空间利用率，并提供了若干附加扩展功能，如访问控制列表（ACL）和文件系统日志。

2.3 Ext2

Ext2 是一个功能强大、易扩充、性能上进行了全面优化的文件系统，各种 Linux 系统都将 Ext2 文件系统作为操作系统的基本部分。

2.4 Ext3

Ext3 是一个日志文件系统，常用于 Linux 操作系统。Stephen Tweedie 在 1999 年 2 月的内核邮件列表中，最早显示了他使用扩展的 Ext2，该文件系统从 2.4.15 版本的内核开始，合并到内核主线中。

2.5 Ext4

Ext4 文件系统是 Linux 系统下的日志文件系统，是 Ext3 系统的增强版，向下兼容 Ext3。下面简单比较这两种文件系统的不同之处：Ext3 目前所支持的文件系统最大 16TB 和文件最大 2TB；而 Ext4 分别支持 1EB 的文件系统，以及 16TB 的文件。Ext4 支持无限数量的子目录，并且引入 Extents 概念，采用多块分配、延迟分配等技术来提高文件系统的性能。Ext4 文件系统还具有在线碎片整理和持久预分配的功能，是对 Ext3 系统一个很好的增强。

有关 Ext2、Ext3 和 Ext4 3 种文件系统的性能对比已经有很多实验，实验大部分在 x86 架构下的传统硬盘下完成，本文后面的测试部分将对这 3 种文件系统在 Android 系统 arm 架构下的表现进行一个完整的测试和考量。

3 Yaffs 文件系统工作机制

在 Yaffs 中，文件是以固定大小的数据块进行存储的，块的大小可以是 512B、1024B 或者 2048B。这种实现依赖于它能够将一个数据块头和每个数据块关联起来。每个文件（包括目录）都有一个数据块头与之相对应，数据块头中保存了 ECC（Error Correction Code）和文件系统的组织信息，用于错误检测和坏块处理。充分考虑了 NAND Flash 的特点，Yaffs 把这个数据块头存储在 Flash 的 16B 备用空间中。当文件系统被挂载时，只须扫描存储器的备用空间就能将文件系统信息读入内存，并且驻留在内存中，不仅加快了文件系统的加载速度，也提高了文件的访问速度，但是增加了内存的消耗。

为了在节省内存的同时提高文件数据块的查找速度，Yaffs 利用更高效的映射结构把文

件位置映射到物理位置。文件的数据段被组织成树形结构，这个树形结构具有 32B 的节点，每个内部节点都包括 8 个指向其他节点的指针，叶节点包括 16 个 2B 的指向物理地址的指针。Yaffs 在文件进行改写时总是先写入新的数据块，然后将旧的数据块从文件中删除。这样即使在修改文件时意外掉电，丢失的也只是这一次修改数据的最小写入单位，从而实现了掉电保护，保证了数据完整性。

结合贪心算法的高效性和随机选择的平均性，Yaffs 实现了兼顾损耗平均和减小系统开销的目的。当满足特定的小概率条件时，就会尝试随机选择一个可回收的页面；而在其他情况下，则使用贪心算法来回收最“脏”的块。

下面从文件操作、物理页的管理、垃圾回收以及安全性四个方面对 Yaffs 进行分析。

3.1 文件操作

Yaffs 的读写过程分为上层的文件操作（创建、修改等）和下层物理空间管理两个部分，其中关于文件的操作主要在源代码中 fs/yaffs2/yaffs_vfs.c 里，该文件中定义了一些 Yaffs 与上层的接口函数；而具体的物理页的分配，则是在 fs/yaffs2/yaffs_guts.c 中，该文件中定义了 Yaffs 的一些核心函数。其中涉及的主要函数和功能如表 1 所示。

表 1 与文件操作相关的函数及其功能

函 数 名	函 数 作 用
yaffs_fill_inode_from_obj()	根据 Yaffs 的结构体 yaffs_obj 来填充 inode 结构体，以便于 VFS 层使用
yaffs_mknod()	创建 obj 及对应的 inode
yaffs_iget()	根据 ino 获得 inode
yaffs_mkdir()	创建文件夹
yaffs_create()	创建文件
yaffs_link()	创建一个新的名为 new_dentry 硬链接，这个新的硬链接指向 dir 目录下名为的 old_dentry 文件
yaffs_symlink()	在某个目录下，为与目录项相关的符号链创建一个新的索引节点
yaffs_lookup()	查找索引节点所在的目录，这个索引节点所对应的文件名就包含在 dentry 目录项中
yaffs_setattr()	设置文件、目录属性
yaffs_getxattr()	获取对象属性
yaffs_removexattr()	消除对象属性
yaffs_find_chunk_cache()	寻找 cache 中的 chunk

3.2 物理页的管理

Yaffs 中有关物理页分配的内容存放于 fs/yaffs2/yaffs_guts.c 中，其中主要涉及了 yaffs_alloc_chunk()函数，该函数实现了分配内存页，该函数通过调用一系列的函数实现了整个内存的分配机制。

每次分配仅使用 block 中的一个页（chunk），其中 dev→alloc_block 为当前正在使用的 block 号，如果 block 号小于 0，则从 Flash 中重新找到一个 block 以供分配。和 chunk 分配不同的是，chunk 的释放在大多数情况下并不释放对应的物理介质，这是因为 NAND 虽然可以按 chunk 写，但只能按 block 擦除，所以物理介质的释放要留到垃圾收集或一个 block

上的所有 page 全部变成空闲的时候才进行。根据应用场合的不同，chunk 的释放方式并不唯一，分别由 yaffs_chunk_del 函数和 yaffs_soft_del_chunk 函数完成。

3.3 Yaffs 文件系统的安全性

Yaffs 文件系统的安全性主要包括 3 个方面的内容：权限控制机制、ECC 校验与坏块标记处理和 checkpoint 机制。

（1）权限控制机制。Android 采用的 Yaffs 文件系统继续沿用 Linux 对文件权限的设置。有关权限的 mode、uid、gid 信息存储在 yaffs_object 中，如图 1 所示。

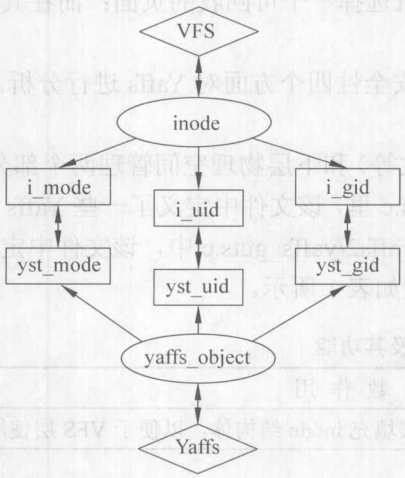

图 1 object 和 inode 中对应权限控制信息

如图 1 所示，yaffs_object 中的 obj→yst_mode、obj→yst_uid 和 obj→yst_gid 分别对应 inode 中的 i_mode、i_uid 和 i_gid。

（2）ECC 校验与坏块标记处理。在 Yaffs 文件系统中，除了存储文件数据以外，还有 oob（out of band）区域。Android 手机的 NAND Flash 在使用过程中不可避免的出现一些坏块，甚至一些 Flash 设备在出厂时已经有一些坏块了。所以在 Yaffs 文件系统中对坏块要做合适的异常处理来避免文件或者数据的丢失。当某 block 的读或者写出现错误时，该 block 即被标记为坏块。怎样判断一个坏块，主要通过 ECC 校验。

（3）checkpoint 作用。除了 ECC 校验外，Yaffs 文件系统中还设置了 checkpoint。checkpoint 中包含了一些关于文件系统的重要数据，当发生错误时可以加快重建运行状态的速度。checkpoint 主要包含的数据有 yaffs_device 信息、block 信息、chunk 标记、文件结构等。

4 演示验证方案和结果分析

为了进一步分析 Yaffs 文件系统性能，我们做了两组实验对 Yaffs 文件系统读写性能进行全面的分析。首先在 nexus s 上完成了 Yaffs 和主流的 Linux 文件系统 Ext2、Ext3、Ext4 的读写性能对比。然后对 Yaffs 和 Ext4、NTFS、FAT32 文件系统在读写小文件小数据和读写大文件大数据时的性能进行对比分析，更全面地了解了 Yaffs 文件系统的性能以及在不同条件下相对于其他文件系统的性能优劣。

下面介绍 Yaffs 文件系统和其他主流文件系统的对比测试情况。

本次测试的目的是反应不同架构相同系统下各文件系统的性能。测试所包含的文件系统有 Yaffs、Ext2、Ext3、Ext4 等主流 Linux 文件系统及 Windows 操作系统下的 FAT32 文件系统。测试的硬件平台包含单核手机、双核手机及四核手机，通过对实验结果的分析来反应文件系统的性能差异。

实验环境如下：

- 硬件平台：Nexus S、Galaxy Nexus、Galaxy S3。

- 目标手机对应系统版本：4.0.1、4.0.4、4.0.4。
- 测试软件：iozone benchmark。

iozone 是一个文件系统的 benchmark 工具，可以测试不同的操作系统中文件系统的读写性能。可以测试 read、write、re-read、re-write、read backwards、read strided、fread、fwrite、random read、pread、mmap、aio_read、aio_write 等不同模式下文件系统的性能。

本次实验是在 arm 架构的手机上进行，首先需要对 iozone 的源码修改后移植到手机上去。在移植完测试工具后，还需要添加不同硬件平台的文件系统支持，如 NTFS、Ext2、Ext3 等文件系统，目前主流的设备上不具有这些文件系统，所以需要将内核支持的开关打开，操作系统就能成功挂载这几种文件系统。当文件系统和测试工具均移植成功后，我们需要在相应的平台下完成预先设定的对比测试实验即可。

我们选取了两种对比实验。

（1）Yaffs 文件系统和 Ext2、Ext3、Ext4 文件系统的读写性能对比实验。

（2）Yaffs 和 Ext4、NTFS、FAT32 文件系统在读写小文件、小数据和读写大文件、大数据时的性能差异实验。

其中实验一的实验结果如图 2 所示。

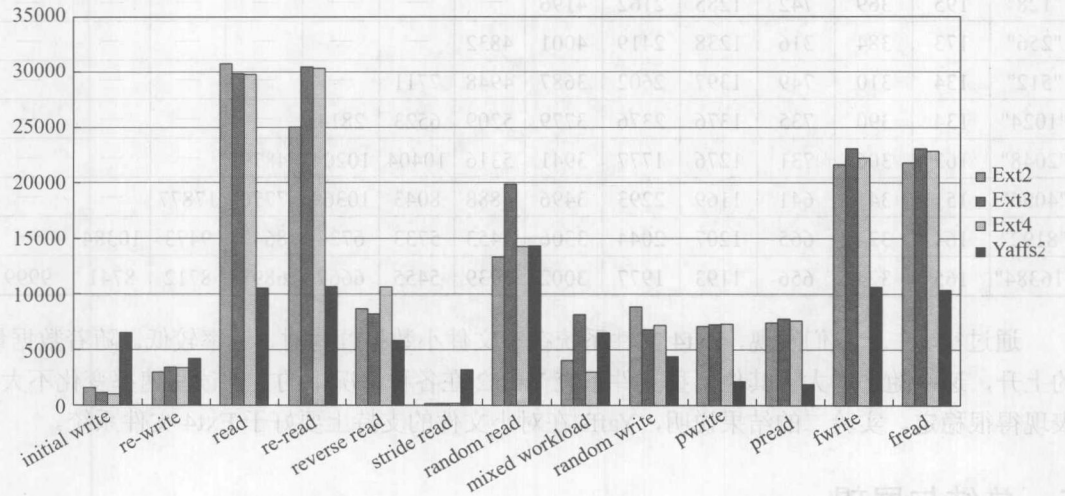

图 2　Yaffs 与 Ext2、Ext3、Ext4 性能对比实验结果

实验一的实验结果表明，Ext2、Ext3、Ext4 性能差异不大，与 Linux 系统上的测试结果不太一致。由于 Ext2 文件系统没有日志，所以在写入性能上要好于 Ext3 和 Ext4。但在重复写入，即 Re-write 性能上，Ext2 的表现就没有 Ext3 和 Ext4 好了。从函数级别的测试 Pwrite、Pread、Fwrite 和 Fread 的结果来看在 Android 平台上，3 种文件系统的表现很平均。Yaffs2 文件系统的读写呈现出的特点与 Ext 类文件系统差别较大，这也符合之前对几种文件系统的对比分析。首先，Yaffs2 文件系统的写入速率要明显快于其他 3 种 Ext 文件系统。但在读取速率上却不及 3 种 Ext 文件系统。在函数级别的测试中，Yaffs2 的速率要比其他 3 种文件低很多。

实验二的结果如表 2 和表 3 所示。

表 3 中横坐标的单位为 kb/s，纵坐标的单位是 kb。

表 2 Yaffs 写入结果

	"4"	"8"	"16"	"32"	"64"	"128"	"256"	"512"	"1024"	"2048"	"4096"	"8192"	"16384"
"64"	5852	5794	5980	6090	5896	—	—	—	—	—	—	—	—
"128"	6309	6208	6593	6255	6458	6355	—	—	—	—	—	—	—
"256"	6259	6656	6690	6681	6704	6676	6745	—	—	—	—	—	—
"512"	6643	6645	6615	6703	6710	6692	6647	6669	—	—	—	—	—
"1024"	6694	6649	4974	6790	6751	6740	6691	6681	6701	—	—	—	—
"2048"	6710	6710	6759	6779	6525	6680	6501	6656	6671	6645	—	—	—
"4096"	6660	6680	6683	6685	6678	6674	6652	6610	6599	6649	6659	—	—
"8192"	6643	6692	6693	6726	6734	6719	6728	6721	6732	6728	6743	6734	—
"16384"	6690	6767	6740	6715	6700	6672	6674	6667	6696	6706	6704	6740	6672

表 3 Ext4 写入结果

	"4"	"8"	"16"	"32"	"64"	"128"	"256"	"512"	"1024"	"2048"	"4096"	"8192"	"16384"
"64"	205	350	769	1579	2283	—	—	—	—	—	—	—	—
"128"	195	369	742	1285	2162	4196	—	—	—	—	—	—	—
"256"	173	384	316	1238	2419	4001	4832	—	—	—	—	—	—
"512"	134	310	749	1397	2602	3687	4948	7711	—	—	—	—	—
"1024"	134	390	735	1376	2376	3779	5209	6523	2814	—	—	—	—
"2048"	162	306	731	1276	1777	3941	5316	10404	10204	4879	—	—	—
"4096"	151	341	641	1169	2293	3496	4888	8043	10369	7770	17877	—	—
"8192"	164	324	665	1207	2044	3306	4453	5733	6724	8642	9473	10384	—
"16384"	165	324	656	1193	1977	3002	3939	5455	6667	6898	8712	8741	9999

通过实验二，我们发现，Ext4 文件系统在小文件小数据读写时，速率较低。随着数据量的上升，速率随之增大。其他两种文件系统 Yaffs2 在各种情况下的文件读写速率变化不大，表现得很稳定。实验二的结果说明，Yaffs 在对小文件的支持上要好于 Ext4 文件系统。

5 总结与展望

本文通过阅读 Yaffs 文件系统源码，分析了 Yaffs 的工作机制。然后又设计了 3 组实验，首先测试了 Yaffs 在不同平台环境下的读写性能。然后对比其他文件系统，对 Yaffs 进行了一个全面的性能测试。通过 Yaffs 和 Ext2、Ext3、Ext4 的读写性能对比实验，发现 Yaffs 文件系统的写入速率明显快于其他 3 种 Ext 文件系统，但在读取速率上却不及 3 种 Ext 文件系统。在函数级别的测试中，Yaffs 的速率要比其他 3 种文件系统低很多。后来对比 Yaffs 和 Ext4、NTFS、FAT32 文件系统在读写小文件、小数据和读写大文件、大数据时的性能差异，发现 Yaffs 和 NTFS 在文件大小不同时读写速率比较稳定，而 Ext4 和 FAT32 在小数据小文件时读写速率较低，随着数据量的上升，速率随之增大。

通过实验，分析了 Yaffs 文件系统的性能以及相较于其他文件系统的优缺点，进一步印证了我们分析其工作机制时对于性能的猜测。在后续的工作中我们可以对实验数据进行

进一步的分析，以期得出更多的结论。同时，我们可以细化性能分析粒度，例如，针对不同盈余空间的情况进行讨论等，更详细、更细致地分析 Yaffs 文件系统的各项性能，同时对其性能中的不足进行优化。

本文主要工作受到核高基重大专项（2012ZX01039-004）和国家高技术研究发展计划（863）（No. 2011AA01A204）项目资助。

参考文献

[1] 毛德操，胡希明. Linux 内核源代码情景分析. 杭州：浙江大学出版社.

[2] 沈建华，罗悦怿. 基于 NAND Flash 的 FFS 设计与实现[J]. 计算机应用与软件，2005（6）.

[3] 罗赟赛，李项军，赵锡凑，等. NAND 闪存在嵌入式 Linux 信息代理中的实现[J]. 仪器仪表用户，2005（5）.

[4] 毛勇强，黄光明. Yaffs 文件系统在嵌入式 Linux 上的实现[J]. 电子设计应用，2006（1）.

[5] 孙天亮，陈伟元，王豪才. 嵌入式系统中 K9F5608U0M NAND 闪速存储器的应用[J]. 单片机与嵌入式系统应用，2002（7-12）.

[6] 吴娴. 一个嵌入式 Linux 文件系统的优化[J]. 计算机应用与软件，2005（7）.

[7] 维基百科. Ext2、Ext3、Ext4 文件系统的介绍 http://zh.wikipedia.org/zh-cn/Extn.

The Analysis and Comparison of Several Common File Systems on Android Platform

Cangzhou Yuan[1], Yuxiao Liu[1], Jian Zhang[2], Lei Wang[2+]

[1] School of Software, Beihang University, Beijing 100191, China

[2] School of Computer Science, Beihang University, Beijing 100191, China

[+] Communication Author: Tel: +8610-8231-6284, E-mail: wanglei@buaa.edu.cn

Key words: File system; Yaffs; performance analysis; Android

Abstract: Yaffs is the first designed specifically for embedded NAND Flash memory file system for large-capacity storage devices, the Android system has been widely used. In this paper, we first analyze the file system Yaffs working principle and performance characteristics, test environments on different platforms Yaffs read and write performance. Then compared to other file systems on Yaffs conduct a comprehensive performance testing. By Yaffs and Ext2, Ext3, Ext4 read and write performance comparison experiments and found that the write speed file system Yaffs significantly faster than the other three Ext file system, but not on the reading rate and three kinds Ext file system. In the function-level tests, Yaffs rate than the other three file systems is much lower. Later contrast Yaffs and Ext4, NTFS, FAT32 file system to read and write small files in a small data file read and write large difference in performance when large data found Yaffs and NTFS file size is not the same in literacy rates relatively stable, while Ext4 and FAT32 in small small data file read and write rates low, as data volumes increase, the rate increases.

Linux 系统调用引发的内存错误检测

徐永健[1,2] 陈 渝[2] 王 丹[1] 肖奇学[2]

[1] 北京工业大学 计算机学院，北京，100124

[2] 清华大学 计算机科学与技术系，北京，100084

摘 要：内存错误是指计算机程序对内存读取与写入、分配与释放、内存泄露等与内存有关的错误，对程序的可靠性有着非常重要的影响。一旦程序在运行期间发生内存错误，则可能导致错误的运算结果或者程序异常退出。对于 Linux 内核来讲，内存错误带来的后果可能更加严重，甚至可以直接造成系统崩溃；所以我们主要针对 Linux 内核中可能发生的内存错误进行分析研究，开发了相应的检测工具。我们的检测工具是在 S2E 的基础上开发的，融合了动态分析、符号执行、具体执行、全系统模拟等技术，可以用于检测 Linux 系统调用所能引发的内存分配和内存使用有关的错误。

关键词：Linux；内核；内存错误；系统调用

1 引言

常见的内存错误主要分为内存读写错误、内存分配与释放错误和内存泄露 3 种类型。其中内存读写错误包括堆栈越界读写、未初始化内存读写、非法指针读写等；内存分配和释放错误包括已释放内存再释放、内存分配未释放、内存分配错误等[1]。另外，缓冲区溢出错误包含堆栈越界访问等错误类型，是安全漏洞的一种普遍形式，也是一个长期存在的安全因素[2]；在对信息系统发动的攻击中，针对缓冲区溢出漏洞进行的攻击已经成为一种重要的攻击方式，CERT 声称 50%以上的攻击都是利用缓冲区溢出漏洞进行的[3]。并且许多（56%）缓冲区溢出的漏洞是由字符串操控函数（如 strcpy、memcpy）的不正确使用引起的[5]。

Linux 内核中发生内存错误可能比用户程序中发生内存错误造成更加严重的结果，此外，对用户程序来说使用系统调用是它能够进入内核的唯一途径。所以我们开发了本文中描述的工具，利用系统调用进入 Linux 内核，之后针对内存的溢出访问和非法分配进行检测。

本文的工作包括：

（1）提出了一种检测内存错误的方法，通过符号化参数的系统调用进入内核，然后基于全系统模拟、具体执行、符号执行和约束求解等技术进行检测。

（2）在 S2E[7]的基础上实现了检测工具原型系统的设计和开发，可以用于检测内存的非法分配和非法访问等错误。

（3）对一个特定的 Linux 内核版本进行了几个网络相关的系统调用的测试，结果发现

资助项目：核高基重大专项经费资助（2012ZX01039-004）

联系作者：徐永健，xuyongjiande@gmail.com

了一个整数溢出造成的内存越界拷贝错误，这证明了本文方法和工具是有效的。

2 相关研究

本文通过以下几个方面来讨论相关领域的工作：

2.1 内存错误

本文主要检测的内存错误为内存分配和内存访问有关的错误，其中最主要的便是缓冲区溢出。Crispin Cowan 在 2000 年对缓冲区溢出进行了研究[8]，文章中指出此前十年中，缓冲区溢出是安全漏洞中最普遍存在的形式，并且在对信息系统安全的攻击中被普遍使用。基于这些原因，近十几年中，许多研究者已经开发了一些缓冲区溢出漏洞的检测工具并且讨论过如何有效防范缓冲区溢出漏洞（BOA）带来的攻击。这些工具，有的是基于源代码的，有的则是基于动态二进制分析和符号执行技术进行检测的。

在近几年中，一些有效的检测工具被开发并且广泛使用，因此内存错误已经大幅减少[4]。但是内存分配错误、指针使用错误等还依然有着很广泛的影响，这也是我们的研究针对内存错误，尤其是内存分配和内存使用方面的原因。

2.2 动态分析

静态源代码分析可以分析出所有可能的路径，但是在处理比较复杂的语言结构时它可能带来非常多的误报[9]；而动态分析则相反，它的检测结果可以非常精确，但是难以发现可以触发漏洞的执行路径。因此，一些研究者开发出了一些动态分析和静态分析相结合的检测工具，比如 Sanjay Rawat 的论文[10]中展示了一个检测 C 语言中缓冲区溢出的方法，它由两个主要的组件构成，一个是静态分析组件（STAC），另一个是依赖污点序列的遗传算法（TAGA）。

本文的方法是让客户机运行在一个被修改过的 QEMU 虚拟机中，并且融合选择性符号执行[11]和约束求解技术。这样测试程序和客户机系统将在具体执行和符号执行两种模式下运行，运行过程受到工具的实时跟踪和检测。由于融合了具体执行和符号执行技术，所以该方法不仅具有准确的检测结果而且可以获得很好的路径覆盖率。

2.3 符号执行

1976 年 James C. King 提出了符号执行技术[12]，之后它便被广泛应用在很多的科学研究和工程项目中，如 KLEE、SPF 和 S2E 等。

KLEE[13]是一个符号执行工具，它能够实现高覆盖率的测试并且可以针对复杂且环境敏感的程序自动生成测试用例。SPF[15]则融合了符号执行、模式识别和约束求解技术，可以为 Java 程序自动检测错误和生成测试用例。S2E 是一个用于分析软件系统属性和行为的测试平台[14]，也可以用于全面的性能分析、私有软件的逆向工程、自动化测试内核模式和用户模式的可执行程序。我们使用了 S2E 作为基础平台，开发了针对内存错误的检测功能。

3 整体架构

图 1 展示了系统的整体架构，首先介绍 S2E 基础平台。S2E 对 QEMU[17]、KLEE 和 LLVM[18]工具链进行了修改和重用，因此融合具体执行、符号执行和约束求解技术。S2E

可以运行在多种系统中，比如 Mac OS X、Microsoft Windows、Linux；并且可以支持多种架构的客户机系统，如 x86、ARM 和 PowerPC。另外，S2E 提供了两类主要接口：选择接口和分析接口；研究中我们使用这些接口开发了内存错误检测工具。

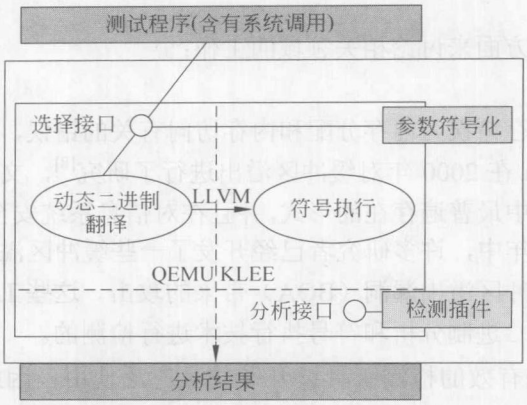

图 1 系统整体架构

本文主要针对了几个网络部分有关的系统调用进行了实验检测，比如 socket、sendmsg 等。

测试程序在两种模式下执行：一种是具体执行，另一种是符号执行；这两种模式下，内存分配和内存使用在 Linux 内核中都会被执行到。当内存分配和使用的操作被监测到正在执行时，检查工具会检测它们的合法性并且把检测结果写入到分析结果报告中以供审核。

4 系统设计与实现

这部分展示工具的详细设计和实现，主要从以下 3 个关键点讲述：测试程序的制作、内存分配的检测、内存使用的检测。

4.1 测试程序的制作

S2E 提供了供用户测试程序使用的头文件，主要包含了一些用于控制分支执行和符号化数据的函数。比如 s2e_enable_forking()用于使多路径执行有效，s2e_make_symbolic()用于符号化用户指定的变量或者数据。

需要注意的是：之所以每次只符号化一个系统调用的参数或者更少（一个系统调用的部分参数），是因为符号执行技术固有的路径爆炸问题。如果同时符号化了很多的调用参数，那么测试的执行时间也将指数级增长。

4.2 内存分配的检测

内核中的一些特定函数负责内存分配，当监控到这些函数执行的时候，工具将检测它们的参数和返回值。通过监控这些函数的起始地址来获知它们什么时候被执行，但是要注意每个内核版本甚至相同内核版本的不同编译选项都会造成这些地址的变化。

通过监控每条指令的地址来判断是否执行到了这些函数，如果是，那么通过栈指针和参数的偏移获取到这些函数的参数，然后进行具体的检测。针对内存分配函数的分配长度，系统进行了主要的检测，流程参考见图 2。