第35卷 第11期 计 算 机 工 程 2009年6月
Vol.35 No.11 Computer Engineering June 2009
·安全技术· 文章编号：1000—3428(2009)11—0131—03 文献标识码：A 中图分类号：TP309

采用 SPIN 的 L4 内存管理形式化验证

陈超超，曾庆凯
(1. 南京大学计算机软件新技术国家重点实验室，南京 210093；
2. 南京大学计算机科学与技术系，南京 210093)

摘 要：模型检验通过状态空间搜索检验一个给定的计算模型是否满足某个用时序逻辑公式表示的特定性质。对 L4 微内核操作系统的内存管理机制进行形式化抽象建模，针对 L4 内核 API 提供的地址空间操作原语 Grant, Map 和 Flush 等操作进行形式化描述，模拟地址页面映射的树形结构管理，运用模型检验工具 SPIN 对抽象模型进行了验证。
关键词：L4 微内核；地址空间操作原语；模型检验

Formal Verification of L4 Memory Management Using SPIN

CHEN Chao-chao, ZENG Qing-kai
(1. Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093;
2. Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

【Abstract】Model checking is a technique that relies on building a finite model of the system and checks whether the desired properties hold in that model. The check is performed as an exhaustive state space search. This paper introduces a model for L4 microkernel memory management system, gives formal description for operations such as Grant, Map, Flush, proposes and verifies some safety properties using SPIN model checker.
【Key words】L4 microkernel; address space primitives; model check

1 概述

安全操作系统的设计、测评及实现过程中对形式化方法有很高的要求。形式化验证的意义在于它能帮助软件开发人员发现其他方法不易发现的系统描述不一致、不明确或不完整[1]。因此，形式化验证是提高软件系统，特别是高等级安全系统的安全性和可靠性的重要手段。

针对当前大部分操作系统软件难以满足对系统可靠性和安全性的要求，微内核的设计理念通过最小化内核，在内核中仅存放那些最基本的核心操作系统功能，实现系统服务的实现与系统基本操作规则的分离，提高系统的可靠性和安全性。L4 是基于最小化、灵活性和效率原则基础上设计的第 2 代微内核，仅包含内存管理原语、线程创建和调度、IPC 和中断处理 4 个方面的功能[2]。在内存管理方面，L4 内核仅仅提供实现内存管理策略的基本操作，允许不同保护策略的实施和用户态的内存管理。

虽然 L4 是一个优秀的微内核的设计，但是距离一个安全操作系统还有许多工作要做。正如 L4 之父 Jochen Liedtke 所说：“仅有微内核并不能真正工作，它并没有强制执行任何安全策略”。L4 设计的目的是着力提高微内核的性能，没有过多地考虑安全问题。因此，需要运用形式化的方法来证明内核功能的正确性和安全性，针对可能的安全性缺陷进一步发展和完善 L4 的设计。

2 相关的验证研究

目前，国外许多研究机构已针对 L4 进行了许多验证研究工作。其中，L4.verified[3]是澳大利亚 NICTA 领导的基于 L4 的安全微内核项目，借助形式化的方法和逻辑演算验证 L4 内核的正确性和可信性。

德国 Dresden 大学在 L4 内核的基础上对安全性着重考虑后新建了一个子项目 L4.sec[4]，其 API 与以往 L4 API 的区别在于加强了通信的控制和内核资源的管理。L4 中的 map, unmap 等内存操作及 I/O 访问权限的基本机制在 L4.sec 中演变成对包括所有内核对象权限的授予与回收，并且作为 L4.sec 中访问控制的基本机制。

通过这些研究可以发现，L4 作为微内核的典型代表已成为安全操作系统研究者关注的焦点之一。一些安全操作系统的设计与开发以 L4 微内核为基础，同时保留了 L4 设计的所有特点和提供的系统调用。L4 在一定程度上已演变成微内核设计标准的代名词。同时，L4 所提供的内核功能可能存在的安全性缺陷使形式化验证成了安全操作系统设计与开发中不可或缺的工作。以内存管理为例，L4 只提供实现内存管理策略的基本操作。实际的内存管理由用户态的服务进程执行。如何保证用户进程以正确的策略映射页面，保证不同用户进程的隔离，实现上层的安全策略，需要运用形式化的方法证明内存管理策略的正确性和安全性。

3 模型检验方法与工具

国际上目前对操作系统的形式化验证主要有 2 类技术，即定理证明和模型检验[5]。定理证明是将要验证的系统及性质用合适的逻辑系统表示为逻辑公式，然后用定理证明器证明性质在系统中是否被满足。定理证明的优点是它适用于无

基金项目：国家自然科学基金资助项目(60773170, 60721002)；国家“863”计划基金资助项目(2006AA01Z432)
作者简介：陈超超(1982－)，男，硕士，主研方向：形式化验证，模型检验；曾庆凯，教授、博士生导师
收稿日期：2008-10-26 **E-mail：**ed_chao@hotmail.com

穷状态系统，缺点是在证明的过程中需要大量的人工干预。如果公式被证伪，定理证明不能提供相关的诊断信息。模型检验是一种关于系统性质验证的方法，通过状态空间搜索的方法检验一个给定的计算模型是否满足某个用时序逻辑公式表示的特定的性质。

对于有穷状态系统，这个问题是可判定的。模型检验技术的优点是自动化程度高，不需要使用者掌握大量的逻辑知识。当所设计的系统不满足某个性质时，模型检验工具可以返回一个反例，通过对反例的解读可以得到性质不成立的原因，为系统的修正提供重要线索。

模型检验可以自动地进行验证在很大程度上归功于有效的软件工具的支持。SMV[6]是美国 CMU 计算机学院开发的模型检验工具，基于“符号模型检验”技术，用以检验一个有限状态系统是否满足 CTL 公式描述的系统性质。其模型检验的基本方法是以二分图 BDDs 表示状态转换关系，以计算不动点的方法检验状态的可达性及其所满足的性质。SPIN[7]是美国贝尔实验室的形式化方法与验证小组开发的模型检验工具，用于检验一个有限状态系统是否满足 LTL 公式描述的系统性质及可达性和循环等一些其他属性。其模型检验的基本方法是以自动机表示各个进程和 LTL 公式，以计算这些自动机的组合与可接受的语言是否为空的方法检验进程模型是否满足给定的性质。

选择 SPIN 是因为 SPIN 允许动态创建并行的进程，并可以在进程之间通过消息通道进行同步和异步通信。L4 内存地址的映射操作是通过进程之间相互发送消息实现的，SPIN 验证主要关心的问题是进程之间的信息能否正确交互，而不是进程内部的具体计算。SPIN 以 Promela 为描述语言，建模方式以进程为单位。一个 SPIN 模型由进程、消息通道、变量和全局对象组成。

4 L4 内存管理模型

L4 的内核提供了简洁高效的地址空间管理原语，负责内存地址空间的映射提供的操作，允许由不同的用户进程以不同的策略映射页面。本文首先描述 L4 地址空间模型，分别给出地址空间操作原语的形式化描述。

4.1 L4 地址空间

L4 内存管理系统提供一系列灵活的分层管理机制来处理地址页面从虚拟地址向物理地址的映射。图 1 表示了 L4 地址空间的分层映射关系。

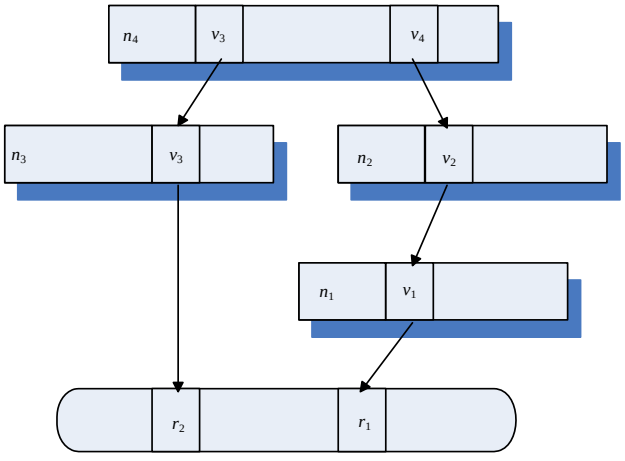

图 1 L4 地址空间映射

图中的矩形框表示各个进程的虚拟地址空间，其中的方形框表示该地址空间中的某个页面，底部的椭圆形框代表了系统的物理地址页面集合，页面之间的箭头代表不同地址空间上页面之间存在的映射关系。

显然，在正确的映射关系下，映射的传递闭包将终止于某个物理地址页面上。如图 1 所示，地址空间 n_1 中的页面 v_1、地址空间 n_2 中的页面 v_2 以及地址空间 n_4 中的页面 v_4 同时映射到物理地址页面 r_1 上。本文用 $R(r_1, r_2, \cdots, r_n)$表示物理页面的集合，用 $V(v_1, v_2, \cdots, v_n)$表示虚拟页面的集合，用 $N(n_1, n_2, \cdots, n_n)$表示地址空间的集合。用户进程地址空间中的每个页面可以用相应的地址空间名和相应的页面号唯一标识为 virtual n v。每个地址空间中的页面或者存在一个对应物理页面的映射，或者是一个空页面。

在形式化模型中，本文用 SPIN 进程的局部变量表示 L4 用户进程的页面管理。地址页面的映射通过 SPIN 进程间的消息通道进行通信。定义消息格式为

```
mtype = {flush, map, grant};
chan ch = [0]of{mtype, int};
```

为了简化验证，设用户进程为 2，物理地址空间大小为 2^2。在 L4 中，系统开始时将创建一个任务 sigma0，并把物理地址分配给它，其他进程的页面都由 sigma0 映射而来。

操作系统的虚拟地址空间管理是由硬件和操作系统内存管理机制共同提供的，每次内存访问都需要完成虚拟地址到物理地址映射的转换。L4 地址空间管理的实现需要一个映射关系库保存这些地址空间之间的映射关系。这个映射关系库借助用户不可见的树型结构实现。系统为每个地址页面维护一棵映射树。例如，Fiasco 以紧凑型数组来实现映射树，数组的大小是 4 的整数次幂(4 ~ 4^15)，数组元素包含节点的深度，按照先根顺序可以重建树的结构。

在形式化模型中，当用户进程数量较少时，可以使用二维邻接矩阵模拟映射树的结构，以 SPIN 全局变量的形式维护映射关系库。初始状态下所有地址页面对应的邻接矩阵元素为 0，即对应的映射树为空。当进程 1~进程 2 存在映射到物理地址 r 的映射关系时，物理地址 r 对应的映射树中加入进程 1~进程 2 的有向边。

本文针对 L4 内存管理建立的 SPIN 形式化模型结构如图 2 所示。

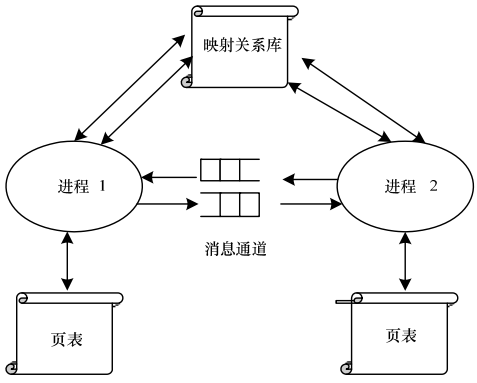

图 2 L4 内存管理 SPIN 模型

4.2 相关操作

L4 内核提供了以下地址空间操作原语：flush,map,grant。flush 用于移除相关地址映射关系，map, grant 用于建立或者转移相关映射关系。

本文用符号 $s \triangleright x \rightarrow^1 y$ 表示在系统状态 s 时存在地址 x

到地址 y 的一个直接映射，其中，y 可以是虚拟地址页面也可以是物理地址页面，但 x 一定是虚拟地址页面。

$$s \rhd x \rightarrow^{1} y = (\exists\, n\ v.\ (x{=}\text{virtual}\ n\ v) \wedge (y{=}\text{virtual}\ n'\ v'\ |\mathrm{r}) \wedge ([v] = y))$$

用符号 $s \rhd x \rightarrow^{*} y$ 代表映射关系的传递闭包。

L4 定义操作 unmap $n\ v$ 为移除所有依赖于 virtual $n\ v$ 的映射关系。unmap 操作对象是执行操作者自己地址空间中的一个页面，作用是把其他地址空间中直接或者间接映射到自己的页面删除，其中，依赖表示的是一种传递映射关系。即

for each m

if $s \rhd m \rightarrow^{*} \text{virtual}\ n\ v$ then $m \rightarrow None$

unmap $n\ v$ 操作将修改该地址页面对应的映射树，将对应节点 n 以下的子树删除。这个操作过程是递归的，unmap 操作发起者通过查映射树向其他地址空间中直接映射到自己的进程发送消息，同时删除映射树中的对应边，消息接收者修改自己的页表，然后递归执行这个过程。

L4 定义操作 flush $n\ v$ 为在 unmap $n\ v$ 操作基础上将地址页面 virtual $n\ v$ 置为 *None*，即

for each m

if $s \rhd m \rightarrow^{*} \text{virtual}\ n\ v$ then $m \rightarrow None$

$n\ v \rightarrow None$

map, grant 操作用于建立新的映射关系。为保证新状态的一致性，必须保证建立的新映射页面能够最终映射到物理页面上。当内核建立一个新的映射关系时，目的地址页面将首先执行刷新操作。而刷新操作可能导致原先映射到该页面的其他地址空间失效。

L4 定义操作 map $n\ v\ n'\ v'$ 为建立映射关系 virtual $n\ v \rightarrow$ ' $n'\ v'$。地址空间 n 将虚拟页面 v 映射到地址空间 n'中的虚拟页面 v'上，v'将首先执行刷新操作，然后更新为新的依赖于 virtual $n\ v$ 的映射。即

if $s \rhd \neg (n\ v \rightarrow None)$ then flush n' v'

$n'\ v' \rightarrow$ virtual $n\ v$

map $n\ v\ n'\ v'$ 操作将修改映射树，更新映射关系。目的地址页面将首先执行刷新操作，再修改自己的页表。

L4 定义操作 grant $n\ v\ n'\ v'$ 为更新目的地址页面，n'v'为源地址页面 $n\ v$ 的映射值，并在源地址页面 $n\ v$ 上执行刷新操作 flush $n\ v$。值得注意的是 grant 操作并非简单地先执行 map 操作再执行 flush 操作。

目的地址页面 n'v'不是映射到源地址页面 $n\ v$，而是映射到源地址页面 $n\ v$ 映射路径上的前一个地址页面。grant 操作在更新映射树时先找到源地址页面对应节点的父亲节点，修改父亲节点到目的地址页面对应节点的映射关系，同时修改目的地址空间页面的映射值，即

if $s \rhd n\ v \rightarrow^{1} m$ then flush $n'\ v'$

$n\ v \rightarrow m \circ$ flush $n\ v$

5 属性验证

SPIN 对抽象模型的正确性分析包括不存在违背断言的状态、不存在死锁、不存在未定义的接收状态以及满足描述系统属性的 LTL 公式。基于第 4 节构建的抽象地址空间模型，可以验证以下一系列安全属性。这些安全属性可以用时序逻辑公式描述为系统的不变量。

属性 1 所有被映射的地址页面都正确映射到物理地址页面上：

$$\Box m, \Box r.\ \ s \rhd\ \ m\ \ \rightarrow^{*} r$$

该属性可以通过检验映射关系的传递闭包是否包含 sigma0 得到验证。

属性 2 虚拟地址到物理地址的映射满足函数映射关系：

$$s \rhd\ \ n\,v \rightarrow^{*} r \wedge s \rhd\ \ n\,v\ \ \rightarrow^{*} r' \Rightarrow r = r'$$

该属性可以通过检验不同物理地址页面对应的映射树是否包含相同进程对应节点的有向边来验证。

属性 3 地址映射路径上不存在环：

$$\forall m,\ s \rhd \neg(m\ \ \rightarrow^{*} m)$$

该属性是系统能够正确进行虚拟地址向物理地址转换的必要条件。在本文构建的原始抽象模型中，该属性验证不能够通过。SPIN 给出的反例说明，当一个用户进程执行 map $n\ v\ n'\ v'$ 操作，而 map 操作目标进程紧接着执行 grant $n\ v\ n'\ v'$ 操作，将导致在 virtual $n\ v$ 上形成环。这个缺陷可能导致在 virtual $n\ v$ 上执行 unmap 或者 flush 操作时递归调用无法终止。可以修改 grant 操作，当目的地址页面位于源地址页面的映射路径上时，不建立新的映射而直接刷新源地址页面，以避免这种情况的发生。修改后的模型可以顺利通过验证。

6 结束语

本文用模型检验的方法对 L4 微内核内存管理机制进行安全验证。针对 L4 内核 API 提供的地址空间操作原语进行了形式化描述，模拟了地址页面映射的树形结构，并运用模型检验工具 SPIN 验证了抽象模型的 3 个基本安全属性。根据 SPIN 给出的验证反例修正了 grant 原语操作可能存在的安全缺陷。

国内外研究机构针对操作系统安全的验证目前主要集中在形式化安全策略模型和形式化规范之间一致性验证，本文对 L4 内存管理机制的建模也是基于 L4 规格说明的。考虑到规格说明和系统实现之间可能存在的差异，未来的工作将关注于系统具体实现和形式化规范之间的一致性证明。

参考文献

[1] Wing J. A Symbiotic Relationship Between Formal Methods and Security[C]//Proceedings of Conf. on Computer Security, Dependability, and Assurance: From Needs to Solutions. [S. l.]: IEEE Press, 1998

[2] Liedtke J. On μ-Kernel Construction[C]//Proceedings of the 15th ACM Symposium on Operating System Principles. New York, USA: ACM Press, 1995.

[3] Derrin P. seL4 Project[Z]. (2008-01-05). http://www.nicta.com.au/.

[4] Volp M, Kauer B. L4.Sec Preliminary Microkernel Reference Manual[Z]. Dresden Technische University, 2005.

[5] Tuch H, Klein G, Heiser G. OS Verification——Now![C]//Proc. of the 10th Workshop on Hot Topics in Operating Systems. [S. l.]: IEEE Press, 2005

[6] McMilan L. Symbolic Model Checking: An Approach to the State Explosion Problem[D]. California, USA: Department of Computer Science, Carnegie Mellon University, 1992.

[7] Holznmnn J. The Model Checker SPIN[J]. IEEE Transactions on Software Engineering, 1997, 23(5): 279-295.

编辑　张正兴