Hyper kernel

这篇文章主要介绍了一种用来保证内核功能正确性的设计实现和验证方法。

Hyperkernel的开发流程分为三个部分，设计，规范化和验证。并且，本文做了如下两个假设，1内核运行在中断禁用的单核处理器上，2内核与用户有着不同的地址空间。

本文的hyperkernel在现有的规范下做了必要的修改来使得验证更加容易，其中值得注意的修改是保证内核接口有限。这样做好处是，如果不这样可能对于大的参数则验证不可扩展，其次，这样的修改不会限制内核态的大小。

接下来是规范化。对于一个有限的接口，开发者需要给出两种规范来描述内核预期的行为，一个是状态状态机规范，另一个是声明性的规范。状态机用来描述系统调用的抽象状态转换，而声明性的规范用来描述高层属性定义。Hyperkernel验证器将会检查状态机规范是否满足声明性规范，这会确保状态机规范的正确性。

C语言在hyperkernel中实现dup函数与在xv6和unix v6中很相似，关键区别是不查找未使用的FD，而是仅仅检查新的fd是否没有使用。这个实现将oldfd所指的文件拷贝到没使用的newfd中去，并增加对应的参考计数器，还会检查两个fd来避免缓冲溢出。

验证阶段。本文给出了两个定理。1内核的实现是状态机规范的改良，2状态机规范满足声明性规范。本文给出了两条定理的证明。定理1保证了验证过的部分不会有低级错误，例如缓冲溢出，除0错误，空指针错误，并且进一步的保证了功能性的正确。定理2保证了状态机规范的正确性。但是，两个定理不能保证内核初始化的正确性。

验证器。首先对内核行为建模，然后将LLVM IR语义翻译成SMT表达式，最后编码crosscutting属性。

为了使内核接口有限，hyperkernel结合了三个办法，1通过硬件虚拟化来处理，2显式的资源管理，3页分类。

局限性。硬件的虚拟化简化了验证难度，这会带来初始化代码会更大和更加复杂；hyperkernel的数据结构是为了高效验证而设计的；hyperkernel需要有限的接口，而很多系统的调用不是有限的；验证器LLVM IR不能扩展到C语言代码和二进制代码；最后，hyperkernel遗留了xv6的一些局限性。