《操作系统H》大作业调研报告

灵活组合的操作系统模块和框架—ArceOS

### 项目背景

在新兴的计算机领域，如偏基础类的机器学习、机密计算，或者偏行业类的自动驾驶、工业机器人等，有各种不同的新应用需求，而现有的传统通用操作系统（如Linux、Windows等）并不能充分满足这些需求。这就推动了这些领域的新玩家加入面向特定领域的新型操作系统的研发中来。但操作系统开发的复杂性让开发者望而却步。我们想解决的问题就是，让开发操作系统变得和开发应用一样方便和高效。

传统的Linux、Windows、FreeBSD其实也采用了模块化/组件化的设计思路，而且支持内核模块的动态加载和卸载。但由于它们把通用性和性能放在第一位，而且采用C语言设计，各个模块之间的数据访问和函数调用都是一种紧耦合的方式。所以，我们可以看到一个特点，这些操作系统的模块很难被其他操作系统使用，即这些操作系统的软件不具有重用性，这样操作系统为了实现对各种应用的支持，且难以重用其他的成熟软件模块，变得越来越庞大和臃肿。

我们认为将来的操作系统不是像现在统治世界的Linux、Windows那样庞大而通用，而是各种可以迅速组合形成的，并且功能丰富多彩的组件化定制操作系统，能够快速适配未来多种多样的处理器、加速器、外设和应用需求，在开发的便捷性、性能和安全性等方面优于已有的通用操作系统。但如何设计组件化定制操作系统是一个需要深入思考的挑战性问题。目前，已有多个项目为尝试采用应用开发模式来开发操作系统内核，这些项目为我们的大作业积累了宝贵的经验。

### 项目描述

基于组件化设计的思路，用Rust语言的丰富语言特征，设计实现不同功能的独立操作系统内核模块和操作系统框架，可形成不同特征/形态/架构的操作系统内核。让领域操作系统易于定制/开发/复用。

### 立项依据

用Rust语言来设计内核组件并形成各种定制的操作系统是一条可行的道路。Rust是一门新兴的编程语言，具有性能高、可靠性强、资源丰富等特点。Rust的运行速度和内存利用率与C语言相仿，在内存安全和线程安全方面远胜于C语言，同时拥有丰富的开发工具、文档和良好的语言生态，是一门理想的系统级开发语言。rCore、zCore、aCore、rCore Tutorial等各种类型的操作系统原型都是用Rust语言开发。同时，ArceOS框架本身也大量使用Rust语言。因此，Rust编程语言是这个项目的一个重要依据。

ArceOS相关的内核组件和框架是这个项目的另一个重要依据。ArceOS是一个rcore的一个子项目，是一个用Rust编写的实验性模块化操作系统，其灵感很大程度上来自Unikraft。ArceOS可以运行在QEMU virt riscv64/aarch64虚拟机上，具有多线程、合作/抢占式调度程序、VirtIO net/blk/gpu drivers、使用 smoltcp 的 TCP 网络堆栈、同步/互斥、具有单个运行队列的SMP调度等功能，集成了helloworld、httpserver等应用，但目前还不支持文件系统、Linux兼容性、中断驱动的设备I/O、异步I/O等功能。ArceOS的优点值得节点，ArceOS的缺点则是我们的项目改进的方向。

|  |  |
| --- | --- |
|  |  |

### 前瞻性和重要性分析

1. **解决操作系统的安全性问题**

主流的通用操作系统和RTOS大部分都是基于C语言开发，C语言简洁高效、灵活性很大，就是为开发UNIX操作系统而诞生的。但随着软件复杂性的增加，即使是操作系统内核的专业开发人员，也会出现常见的内存相关和并发相关的编程错误。因此，采用Rust语言开发操作系统可以最好的平衡系统的安全性和效率，是一种合适选择。

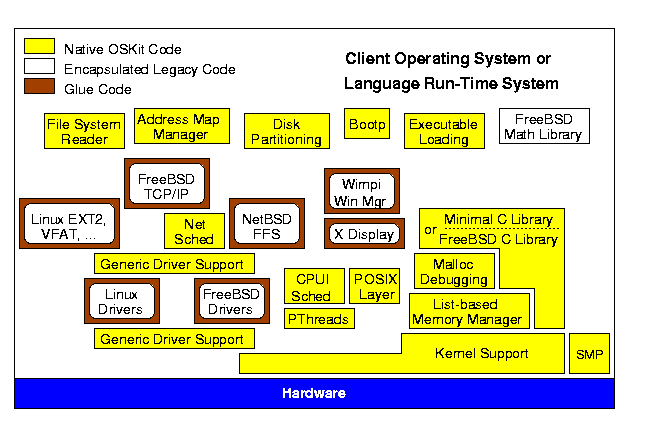
1. **创新操作系统的开发方式**

编写操作系统软件过程繁琐，细节较多，根本原因在于操作系统内部模块广泛的相互依赖带来的软件复杂性，以及操作系统的自包含性带来的软件不可重用性。在传统操作系统的设计实现中，并没有采用层次化的开发方法。以Linux为例，如果要对其中一个模块进行重新设计，几乎需要了解与其它各个模块的依赖关系，这使得Linux内核开发是一个非常有挑战的工作。采用组件化的思路设计操作系统可以有效地解决上述问题，为操作系统的开发提供新的思路。

### 相关工作

1. **OSKit项目**

早在2002年，犹它大学计算机科学系FLUX研究组提出的[OSKit项目](https://www.cs.utah.edu/flux/oskit/)，基于C语言设计了34个组件库和一个连接这些组件库的框架来形成不同功能的特定操作系统内核，其目标在于降低操作系统的研发门槛和成本。目前该项目已经停止维护。



1. **OCaml语言与****[MirageOS](https://mirage.io/)**

MirageOS是一个基于OCaml语言编写的库操作系统，它构建了安全的，高性能和资源节约型unikernel。代码可以在普通操作系统（如Linux或macOS）上开发，然后编译成在Xen或KVM虚拟机管理程序下运行的完全独立的专用单内核。该系统并没有获得广泛的应用，最大的败笔就是使用了错误的编程语言。Ocaml语言的抽象能力很强，安全性也很好，但它具有主流函数式语言的通病，相对于C语言，其通用性很弱，而性能更差，缺少对已有应用的支持，难以得到产业界认可。



1. **Theseus系统**

2018年，Rice大学提出了基于Rust语言的Theseus操作系统，通过精心设计的内核组件，支持操作系统的组件在线更新，使得操作系统的可靠性得到很大增强。Theseus用于试验新颖的操作系统结构、更好的状态管理以及如何利用语言内设计原则将资源管理等操作系统职责转移到编译器中，预期在将在高端嵌入式系统或边缘数据中心环境中发挥作用。但由于过于注重高可靠设计，在性能和已有应用的支持比较差。

1. **Unikraft系统**

2014年以后，多家单位合作设计了Unikraft操作系统。Unikraft是一个快速，安全和开源的Unikernel 开发工具包，采用C语言编写，由不同的组件库构成，支持云计算环境中的常用Linux应用。Unikraft可以根据应用程序的特定需求定制操作系统、库和配置，大幅减少存储空间和其他软件的恶意攻击，提供出色的性能。其unikernel架构设计让应用程序和OS内核都运行在特权态，提高了整体性能。Unikraft的性能和对Linux应用的支持，使得它得到了产业界一定的认可和使用。但Unikraft采用C语言设计实现，这导致了内核组件的重用性和安全性还需提高。



1. **Rcore系统**

Rcore是清华大学uCore OS Plus的Rust版本，将成为下一代教学操作系统。Rcore兼容Linux，支持 x86\_64、RISCV32/64、AArch64与MIPS32平台，具有Linux 系统调用、网络堆栈、文件系统、信号系统、异步 IO等多项功能。此外，Rcore拥有丰富的文档支持（例如rcore-tutorial-book、csdn博客、知乎博客等等），适合初学者学习如何从用Rust从零开始写一个操作系统。