 DOI: https://doi.org/10.12345/jxffcxysj.v5i3.10890

# Design and Implementation of the Class Operating System Based on UEFI

**Qihong Ren Hui Huang Yongliang Wang Hui Zhu**

School of Computer Science and Engineering, Sanjiang University, Nanjing, Jiangsu, 210012, China

**Abstract**

For several G problems of large operating system packages, such as different types of Windows10 are about 3G to 4G; Ubuntu20.10 desktop package is about 3G; This paper discusses the architecture design, file naming design and code development based on UEFI design, and finally the prototype UEFI based on corresponding EDKII source code, code quantity 4K, the post-compiled binary 233K experiment shows that the system can operate.

**Keywords**

operating system; UEFI; architecture; EDKII

# 基于 UEFI 的类操作系统设计与实现

任启红 黄辉 王永亮 朱慧

三江学院计算机科学与工程学院，中国·江苏 南京 210012

**摘 要**

针对当前操作系统非常庞大安装包动辄几个G的问题，如Windows10不同类型的安装包大概在3G到4G；Ubuntu20.10桌面版安装包大概3G，这样的安装包并不小；论文讨论基于UEFI设计与实现类操作系统的架构设计、文件命名设计、代码开发，最后基于UEFI对应的EDKII源码开发原型的类操作系统，代码量4K，编译后二进制233K实验结果表明系统能运行。

**关键词**

操作系统；UEFI；架构；EDKII

## 1 引言

当前市面上 Windows 操作系统在桌面占用基本上一家独大，接着是 Mac OS，最后是 Linux，而 UEFI 自从 1998 年 Intel 公司研发到 2005 年交由 UEFI 论坛推广与发展后，目前市面上新销售的笔记本、台式电脑基本初始化默认自带 UEFI。UEFI 是统一可扩展固件接口，用来定义操作系统与系统固件之间的软件界面，UEFI 用于替代传统 BIOS，并且 UEFI 为上一层操作系统提供非常丰富的接口。EDKII 全称 EFI Developer Kit，它实际上是一套实现了 UEFI 标准的开源代码，开发者可以在此基础上开发 UEFI 下的设备驱动或者其他应用。

## 2 操作系统

操作系统用于管理计算机硬件，如：CPU、内存、硬盘、显示器、鼠标、键盘等等。基于传统 BIOS 的操作系统实现需要写汇编代码、驱动、模式切换等等，基于 UEFI 则可以省去不少工作量，直接调用底层接口，自己实现上层的逻辑层。

【作者简介】任启红（1986–），男，中国四川南充人，硕士，工程师，从事软件工程研究。

## 3 架构设计

整个系统采用四层设计：硬件层、UEFI 层、核心逻辑管理层、图形与事件管理层，如图 1 所示。


图 1 系统整体架构

其中硬件和 UEFI 层直接使用 EDKII 源码，在核心逻辑层对 UEFI 实现的驱动做了一层封装，核心层为左侧虚线方框所示，其中右侧 UEFI 和硬件为系统依赖的下层平台如图 2 所示。

系统源代码文件命名规范：

①语法格式 Ln_MouduleName_SubMouduleName.*，例如：L2_DEVICE_Keyboard.c，L2 表示是第二层，L1 表示第一层，依次类推：

第二个 DEVICE 表示是 DEVICE 模块，其中第二级模块命名全大写，第三个 Keyboard 表示子模块，子模块单词第一个字母大写，若没有子模块命名格式：Ln_MouduleName.*，例如：L2_GRAPHICS.c，表示是 Graphics 根目录下的第 2 层 c 文件，若是头文件则命名为 L2_GRAPHICS.h.

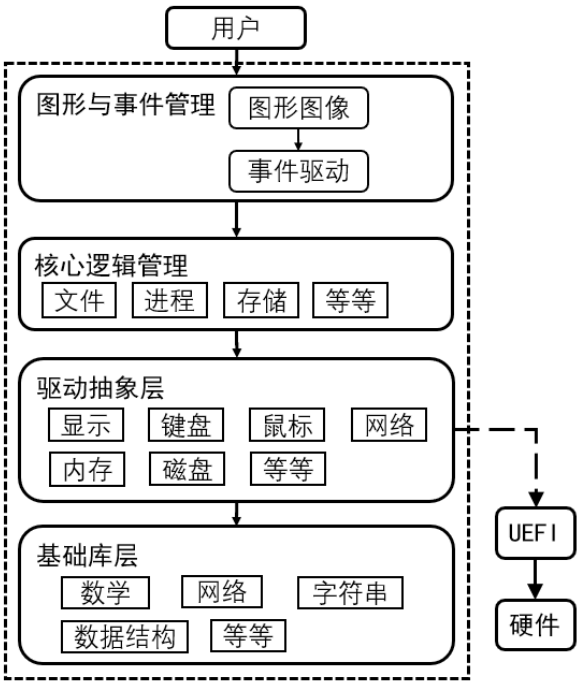


**图 2 系统详细架构**

②最前面的 Ln 其中的 n 表示模块内部的分层，当前架构要求为 L1 为本模块基础数据结构，对其他任何模块不依赖。

③ Ln 其中 n 建议最大到 9，所有模块可以自己确定对外提供的接口列表，默认为一个模块最大层对应的头文件。

④模块内部 Lm 调用 Ln，必须要求 m 大于等于 n，即模块内部同层函数可以调用，或是调用层数数字更小层级；数字小的层不允许调用数字大的层，防止模块内部产生循环依赖环。

源代码目录结构如下：

./Libraries

./Libraries/Network

./Libraries/Math

./Libraries/String

./Libraries/DataStructure

./Libraries/Memory

./Processes

./Devices

./Devices/Screen

./Devices/Mouse

./Devices/Timer

./Devices/System

./Devices/Store

./Devices/Keyboard

./Global

./Memory

./Graphics

./Partitions

./Partitions/NTFS

./Partitions/FAT32

其中：Libraries 是基础库目录，Processes 是进程管理目录，Devices 是设备管理目录，包含一些设备对底层接口的封装，Memory 是内存管理目录，Graphics 是图形图像目录和事件驱动目录，Partitions 是分区管理目录，包含 FAT32 和 NTFS 文件系统的解析，如果后续需要对其他文件系统类型扩充也可以放到这个目录，Global 目录是定义一些全局变量。

## 4 系统实现

在系统实现层面：①多进程采用 EDKII 提供的 gBS->CreateEventEx 事件组接口；②外部存储读取使用 gEfiDiskIoProtocolGuid、gEfiBlockIoProtocolGuid、gEfiDiskIoProtocolGuid 协议；③键盘读取使用到 gEfiSimpleTextInputExProtocolGuid 协议；④鼠标读取使用到 gEfiSimplePointerProtocolGuid 协议；⑤定时器使用到 gBS->SetTimer 接口；⑥屏幕图形显示使用到 gEfiGraphicsOutputProtocolGuid 协议，等。

文件系统是基于外部存储先读取接口读取扇区，再分析扇区内容，最后读取分区的根目录、根目录下的文件、目录，再递归，其中分区读取取和文件读取使用到状态机。

图形与事件处理层，主要是根据用户的鼠标光标在屏幕上位置和点击以及用户键盘输入，出发的事件进行处理，并调用下层提供的接口，使用不少状态机，例如：开始菜单、我的电脑窗口、系统设置窗口等等。

## 5 结语

当前系统在实现和测试的过程中，发现一些问题，如：① UEFI 中断这块支持不太友好，如果要实现操作系统，对运行于操作系统上的应用开发有较大影响；② UEFI 在不同电脑上运行存在兼容性问题，在笔者 HP 笔记本运行正常，

在 HP 台式机运行有问题，在 DELL 笔记本运行也有问题；③如果基于 UEFI 开发，不能执行 CoreExitBootServices，因为这样有些接口不能继续使用，导致系统无法继续使用，也会导致有些内存不能使用；④由于使用 UEFI 提供的接口，很多底层的实现需要花费大量时间去看协议，源码。由于在实现的时候还有很多业务异常分支考虑的不太详细，系统会有些缺陷，但是整个系统是可以运行的，并且二进制比较小，开机大概 1.5 秒的样子。

## 参考文献

[1] 陈瑶,尹玉瀚.计算机新型启动引导UEFI架构分析[J].网络信息工程,2021(8):81-84.

[2] 韩新军,白茹,田玉平.UEFIBIOS系统浅谈[J].企业技术开发,2016,35(15):70-71.