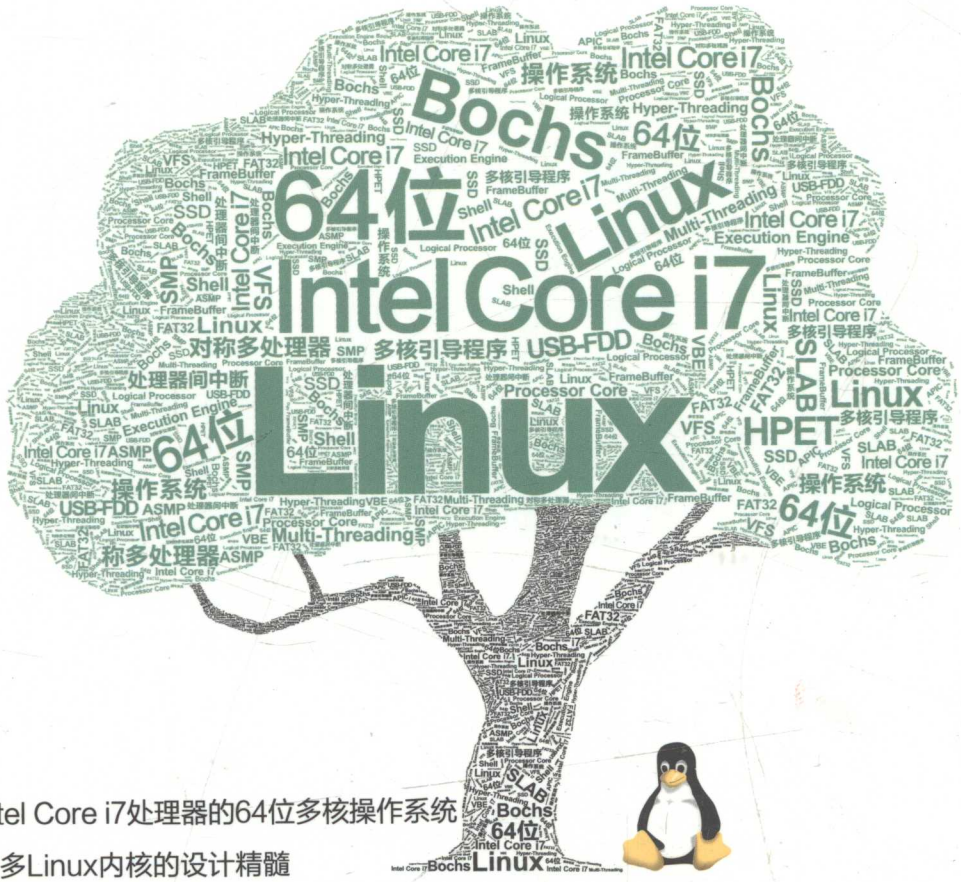
TURING 图灵原创

一个64位操作系统的设计与实现

田宇◎著



- 基于Intel Core i7处理器的64位多核操作系统
- 引入诸多Linux内核的设计精髓
- 既可在Bochs虚拟机中执行，又可通过U盘引导运行于笔记本电脑

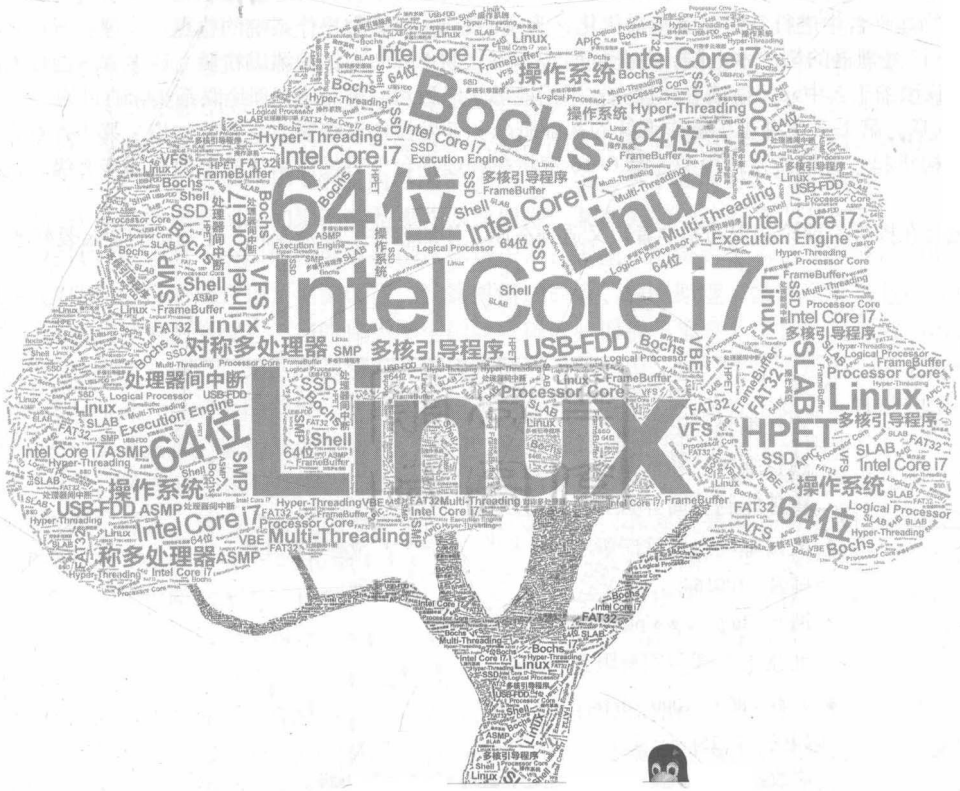
中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原创

一个64位操作系统的设计与实现

田宇◎著



人民邮电出版社
北 京

图书在版编目（CIP）数据

一个64位操作系统的设计与实现 / 田宇著. -- 北京：人民邮电出版社，2018.5
（图灵原创）
ISBN 978-7-115-47525-1

Ⅰ. ①一… Ⅱ. ①田… Ⅲ. ①Linux操作系统 Ⅳ. ①TP316.85

中国版本图书馆CIP数据核字(2017)第316719号

内 容 提 要

本书讲述了一个64位多核操作系统的自制过程。首先从虚拟平台构筑起一个基础框架，随后再将基础框架移植到物理平台中进行升级、完善与优化。为了凸显64位多核操作系统的特点，物理平台选用搭载着Intel Core i7处理器的笔记本电脑。与此同时，本书还将Linux内核的源码精髓、诸多官方白皮书以及多款常用协议浓缩于其中，可使读者在读完本书后能够学以致用，进而达到理论联系实际的目的。

全书共16章。第1~2章讲述了操作系统的基础概念和开发操作系统需要掌握的知识；第3~5章在虚拟平台下快速构建起一个操作系统模型；第6~16章将在物理平台下对操作系统模型做进一步升级、优化和完善。

本书既适合在校学习理论知识的初学者，又适合在职工作的软件工程师或有一定基础的业余爱好者。

◆ 著　　田　宇
责任编辑　王军花
执行编辑　陈兴璐
责任印制　周昇亮

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京市艺辉印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：43.25
字数：1 022千字　　2018年5月第1版
印数：1－3 000册　　2018年5月北京第1次印刷

定价：139.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

前　言

这不是一本由几万行代码简单罗列成的书，也不是一本由各种技术文档堆砌成的书。当你在学习计算机操作系统原理时迷失了方向，它会为你点亮一盏灯，照亮前方的路。

计算机相关专业的读者们在大学时都学习过《操作系统》这门课程。对于什么是操作系统，老师们普遍以理论概念为主进行教授，比如，什么是进程，什么是线程，什么是文件系统等知识点。可是，像进程与线程的创建过程、空间换时间的应用场景等内容却鲜有提及。以上这些问题，我在学生时代的时候特别想弄清楚，但却无从着手，就算有些思路，也因为学艺不精，半途而废了。即使向老师们请教，也只得到理论性的解释，无法获得清晰、准确、具体的答案。我想，也许正在阅读本书的你们也难以将其缘由娓娓道来。不过，可能有些人觉得没有必要非常清楚这些问题，以前的我也曾有过此种想法。待到有幸从事几年 Linux 内核级的研发工作后，我才逐渐对上述问题有了比较直观、深刻的认知，并且慢慢体会到，如果不清楚操作系统原理，某些问题解决起来非常困难。

在计算机领域，中国的发展速度仍然落后于发达国家，师资力量不足是在所难免的。一些学校只是概括性地传授微机原理、汇编语言、计算机组成原理、编译原理、操作系统等基础知识，甚至还可能只将它们作为选修课程。当时作为学生的我觉得这些课程不重要，没有认真细致学习，但在工作多年的反思中才发现，它们是融会贯通计算机领域的必要知识，它们往往决定了一个人能在计算机行业走多远。而且，目前中国软件行业仍以外包为主，能够静下心来做技术储备、基础知识培训的自主研发型公司少之又少，这种局面使得我们想在工作中弥补基础知识依然十分困难。

现今，网络上已有不少关于操作系统实践类的文章和图书，这些文章和图书作为入门学习是很不错的选择。可是，这些文章和图书内容的一个通病是，操作系统普遍采用 Intel 32 位处理器的虚拟平台进行开发、研制。这个 32 位处理器的虚拟平台虽然学起来简单，但如果用到工作中举一反三的话，还是存在诸多差距与不足。比较典型的例子有，虚拟平台与物理平台在软硬件执行流程上的差异、多核处理器间的通信机制、高级中断控制器的配置、先进的 64 位处理器体系结构等，这些问题难以正确分析、推理及解决，会导致理论与实践脱节。

出于以上种种原因，作者想通过一系列图书把现代操作系统的真实面貌展现给读者，并希望借此寻找有兴趣和有能力的朋友们一起开发这款操作系统。考虑到对操作系统感兴趣的读者不在少数，基础知识的掌握水平势必参差不齐。为了照顾到各个方面，本书将尽量做到既适合在校学习理论知识的初学者，又适合在职工作的软件工程师或有一定基础的业余爱好者。

这是一个基于 Intel 处理器 IA-32e 体系结构编写的操作系统雏形。虽说它只是一个操作系统雏形，但为了使读者能够在学习 Linux 内核源代码时得到一些助力，本系统还将 Linux 内核的精髓（提炼自多版 Linux 内核）融入其中，并以物理平台作为主要运行环境、虚拟平台作为辅助运行环境。IA-32e

体系结构可以通俗理解为“64 位处理器”。阅读过 Intel 技术文档的读者应该知道，64 位处理器是在原有 32 位处理器的基础上扩展而得的，其对 32 位处理器的运算速度、数据带宽乃至运行时的高效性、安全性都进行了全面升级与优化。因此，64 位操作系统比 32 位操作系统“快”是有诸多理由的，这些理由很难用几句话解释清楚。

本书从计算机上电启动开始，循序渐进地实现了一个 64 位操作系统的雏形。先介绍一下本书操作系统的硬件运行环境。作者使用 Lenovo ThinkPad X220T 笔记本电脑作为操作系统的物理平台，其上搭载着一枚 Intel (R) Core (TM) i7-2620 M CPU @ 2.70 GHz（这串字符将在正文里通过程序从处理器中取得）双核四线程处理器，并配有 8 GB 容量的物理内存。如果条件允许的话，读者还可以准备一台电脑作为编译环境，否则反复重启同一台电脑会严重影响开发效率。除此之外，还必须准备一个 U 盘，要在物理平台下运行操作系统，怎能少得了 U 盘引导呢！U 盘的容量无需太大，16 MB 或 8 MB 足矣。

时至今日，我依然能够清晰回忆起当初编写这个操作系统时遇到的各种困难，以及经历过的一次次挫败、煎熬与崩溃。此刻，当你们读到本书时，说明那些困难已经成为历史。“失败不可怕，害怕失败才真正可怕。当你意识到失败只不过是弯路时，你就已经走在成功的直道上了。”用这句话，希望可以与读者共勉。

结伴冒险即将开始，希望读者能和作者痛快走一回。

阅读指导

鉴于本书采用迭代方式循序渐进地去实现一个操作系统，而并非一次性构建起来，所以在开发的每个环节都会对之前的代码进行修改、调整和升级。为了节省篇幅，本书会附赠源码和运行效果图，而书中内容将主要针对有变动的重要信息进行讲解，请读者借助代码比较工具（如 Beyond Compare）和运行效果图并行研习。

限于篇幅，本书只对研发期间所涉及的知识进行讲解。对于读者在实践过程中提出的疑问或困惑，本书会略过，还请读者查阅相关官方文档。

对于没有操作系统开发经验和缺乏设计思路的读者，强烈建议你们在阅读完本书后，再按照书中的描述去实践自己的操作系统，以免初次阅读本书时编写出运行效果不佳的程序。

本系统使用的编译器要求汇编代码使用小写字母书写，而 Intel 官方白皮书对汇编指令和寄存器的描述均采用大写字母书写。此种现象源于各个编译器对汇编指令的书写格式要求略有不同，有的编译器甚至会通过前/后缀符号对汇编指令做进一步修饰。为了区分正文中的汇编代码和汇编指令，本书统一使用小写字母表示汇编代码，而汇编指令和寄存器则统一使用大写字母表示。

保留英文缩写

为了做到原汁原味，对于页管理机制（32/64 位处理器体系结构）中常用的专有名词，在不引起困惑的前提下，本书尽量使用英文缩写（更多英文缩写请见“术语表”）。

- PML4（Page Map Level 4）：4 级页表。
- PML4E（PML4-Entry）：PML4 页表项。
- PDPT（Page Directory Pointer Table）：页目录指针表。

- PDPTE（PDPT-Entry）：PDPT 页表项。
- PDT（Page Directory Table）：页目录表。
- PDE（PDT-Entry）：PDT 页表项。
- PT（Page Table）：页表。
- PTE（PT-Entry）：PT 页表项。

鸣谢

本书历经近三年时间撰写而成。在写作期间，作者收获颇多，也沉淀许多。在此，由衷感谢人民邮电出版社图灵公司给予的出版机会，感谢策划编辑王军花、陈兴璐严谨、专注的工作使本书绽放出更加绚丽的光彩，感谢在幕后为本书辛勤付出的编辑们。同时也感谢宋玉鹏先生为本书提供的写作建议与部分插图。

我们在求知的道路上会遇到许多良师益友，前行的每一步都站在巨人的肩膀上，在此特别感谢他们曾经提供的技术帮助与支持。他们是符田、吴昊、王喆、王航、张轶伦、高乐乐、孙海鲛、康思为特、郜弘毅、郜弘睿、李海涛、张鹏、孙林、李麟、化松收、赵晓燕、崔鹏、乔国荣、姜峰、赵云云、周海龙、刘永康、崔永、毛振宇、苏立斌、张超、佘建伟、张松、刘昊、李庆松、曹美玲、王超、杨沐天、杨晗、赵兵、甄帅。

今天之所以不同于昨天，恰恰是因为昨天的感受依然在我们心中。

目　录

第三部分 高级篇

请从这里开始你的旅程……

第一部分

操作系统相关知识介绍及环境搭建

这一部分将介绍操作系统相关知识及环境搭建方法，包含两章内容：

- 第 1 章 操作系统概述；
- 第 2 章 环境搭建及基础知识。

本部分主要针对业余爱好者以及一些基础知识薄弱的朋友们。如果你是从事底层开发工作的软件工程师，或是具有扎实的知识基础和编程功底的业余爱好者，则可以快速阅读这部分内容，或者选择跳过。

第1章 操作系统概述

本章首先从宏观上介绍操作系统由哪几部分组成，然后介绍编写操作系统必须掌握的知识，最后再简要介绍本书操作系统。

1.1 什么是操作系统

操作系统这个概念非常宽泛，不论是办公、生活使用的设备与计算机，还是机械工业生产制造使用的仪器仪表，它们都装有操作系统。哪怕是只有一条指令的单片机，也可以称作嵌入式操作系统。从这个宏观意义上出发，操作系统和硬件设备就可以区分开，它们要么是只有硬件电路的裸机，要么是含有操作系统的硬件电路。（这里的硬件电路指含有处理器的可编程电路。）

对于只有一条指令的单片机来说，它被称为操作系统，未免显得太过牵强。在大多数人眼里，操作系统应该由功能强大高效运转的核心、万能的驱动程序、绚丽的操作界面、舒适简洁的操作方式及方便实用的工具组成。可在当年，操作系统却是一个连硬盘都没有而只有一些简单逻辑门电路的怪物。从操作系统的发展史来看，它经历了单任务系统、批处理系统、分时操作系统、实时操作系统、嵌入式操作系统以及时下最流行的云系统等阶段。随着时代的发展，硬件在不断更新换代，操作系统也在不断演化，操作系统的功能会因为应用场景不同而具有不同的特点，但它的根本目的依然是为了方便人们对硬件设备的使用与交互。

1.2 操作系统的组成结构

一款功能完备、方便易用的操作系统，是由一套庞大的结构组成的，图1-1描述了操作系统的整体结构。

从图1-1可以看出，操作系统由内核层与应用层两部分组成。内核层主要由引导启动、内存管理、异常/中断处理、进程管理、设备驱动、文件系统等模块组成，而系统API库和应用程序则属于应用层的范畴。之所以将内核层和应用层分开，是因为内核层主要负责控制硬件设备、分配系统资源、为应用层提供健全的接口支持、保证应用程序正常稳定运行等全局性工作。而应用层主要负责的是人机交互工作。下面将对各个模型逐一进行介绍。

- **引导启动**。引导启动是指，计算机从BIOS上电自检后到跳转至内核程序执行前这一期间执行的一段或几段程序。这些程序主要用于检测计算机硬件，并配置内核运行时所需的参数，然后再把检测信息和参数提交给内核解析，内核会在解析数据的同时对自身进行配置。如图1-1所

示，使用横线将引导启动模块与其他内核层模块分隔开，是考虑到引导启动模块只是为了辅助内核启动，而并非真正属于内核。一旦内核开始执行后，引导程序便再无他用。如果把内核比作卫星的话，那么引导程序就相当于运载火箭，卫星进入轨道后，火箭就完成了它的使命。引导启动程序曾经分为两部分——Boot和Loader，现在通常把两者的功能合二为一，并统称为BootLoader。

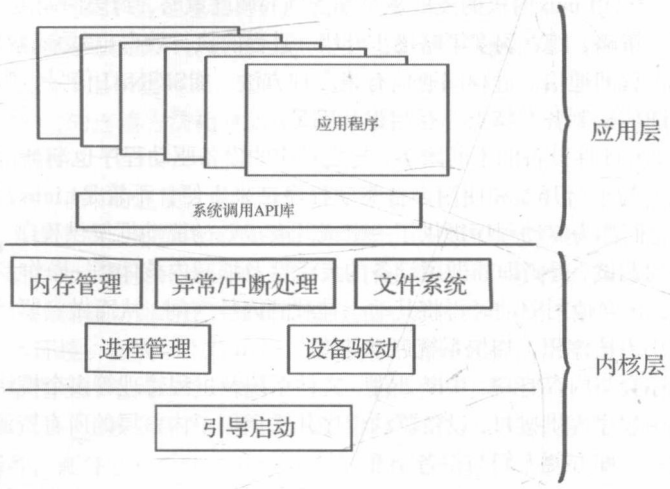


图1-1 操作系统整体结构图

目前，比较流行的引导启动程序有Grub和Uboot等，它们的功能都比较强大，用户可以通过它们自带的终端命令行与之进行简单的交互，此举为控制内核的加载和使用提供了诸多遍历。

- **内存管理**。内存管理单元是内核的基础功能，它的主要作用是有效管理物理内存，这样可以简化其他模块开辟内存空间（连续的或非连续的）的过程，为页表映射和地址变换提供配套函数。Linux内存管理单元的伙伴算法，算是一种稳定成熟的内存管理算法，它可以长时间保持内存的稳定分配，防止内存碎片过多。还有内存线性地址空间的红黑树管理算法，它将原有的线性地址结构转换为树状结构以缩短搜索时间，同时又在每次插入新节点时调整树的高度（或者深度），来维持树的形状进而保证搜索时间的相对稳定，该算法既兼顾搜索时间损耗又兼顾插入时间损耗。因此，Linux选择红黑树这种近似平衡树来代替之前的AVL树（绝对平衡树）也是出于这方面的考虑。
- **异常/中断处理**。此处的异常是指处理器在执行程序时产生的错误或者问题，比如除零、段溢出、页错误、无效指令、调试错误等。有的异常经过处理后，程序仍可继续执行，有的则不能继续执行，必须根据错误类型和程序逻辑进行相应的处理。而中断处理是指处理器接收到硬件设备发来的中断请求信号并作出相应处理操作。这部分内容与外围硬件设备关系非常密切，它的处理效率会影响操作系统整体的执行速度。通常，中断处理会被分为中断上半部和中断下半部。中断上半部要求快速响应中断，在取得必要的数据和信息后尽早开启中断，以使处理器能够再次接收中断请求信号。中断下半部被用来执行剩余中断内容，像数据解析、驱动程序状态调整等更耗时的内容均在这里完成。为了让更紧迫的进程优先执行，中断下半部还可将处理内

容安放在一个进程中，以让更高优先级的进程得到快速执行。

- **进程管理**。说到进程，想必会有人疑惑进程和程序的区别。程序是静静地躺在文件系统里的二进制代码，属于静止状态。一旦把这个程序加载到操作系统内运行，它就变成了进程。进程是程序的运行状态，所以它会比程序拥有更多管理层面的信息和数据。

  说到进程管理功能，不得不提进程调度策略，一个好的进程调度策略，会提高程序的执行效率和反应速度。现代Linux内核的发展从早期的O(1)调度策略，到楼梯调度策略，再到现在的CFS完全公平调度策略，随着调度策略逐步升级，进程的执行效率也越来越高。进程管理的另一个重要部分是进程间通信。进程间通信有很多种方法，如SIGNAL信号、管道、共享内存、信号量等，这些通信机制各有特点，互相弥补不足。
- **设备驱动**。随着硬件设备的不断增多，与之对应的设备驱动程序也渐渐占据了操作系统的很大一部分空间。为了给开发和使用设备驱动程序带来方便，不管是Linux操作系统还是Windows操作系统，它们都为驱动程序提供了一套或几套成熟的驱动框架供程序员使用。同时，为了便于驱动程序的调试、提高即插即用设备的灵活性及缩减内核体积，操作系统逐渐把驱动程序从内核中移出，仅当使用驱动时再将其动态挂载到内核空间，从而做到驱动程序即插即用。这样一来大大缩小内核体积，加快系统启动速度。

  设备驱动程序会与内存管理、中断处理、文件系统及进程管理等多个模块共同协作。为了让硬件设备给应用程序提供接口，设备驱动程序几乎调用了内核层的所有资源。这也是开发操作系统的目的之一，即方便人们与设备交互。
- **文件系统**。文件系统用于把机械硬盘的部分或全部扇区组织成一个便于管理的结构化单元。此处的扇区也可以是内存块，这样便组成了一个RAMDisk（内存式硬盘）。这样一个内存式硬盘单单在文件读写速度上就比普通机械硬盘高出一个数量级，其显而易见的缺点是掉电后数据全部丢失。不过与它的优点相比，这个缺点是完全可以忍受的，比如Linux内核的sys文件系统便是在RAMDisk中创建的。

  文件系统的种类也是纷繁复杂的，像上面提到的sys文件系统，还有大家耳熟能详的FAT类文件系统，以及Linux的EXT类文件系统，它们对扇区的组织形式虽各具特色，却都是为了给原生操作系统提供方便、快捷的使用体验而设计的。
- **系统调用API库**。系统调用API库接口有很多规范标准，比如Linux兼容的POSIX规范标准。对于不同的接口标准来说，其定义和封装的函数实现是不一样的。不管怎么说，系统调用API库最终都是为了给应用程序提供简单、快捷、便于使用的接口。
- **应用程序**。应用程序包括我们自己安装的软件和系统提供的工具、软件与服务。

  在众多应用程序中，比较特殊的一个应用程序是系统的窗口管理器，它主要用于管理图形界面的窗口，具体包括窗口的位置布局、鼠标键盘的消息投递、活动窗口仲裁等功能。
  - 窗口的位置布局负责控制窗口的比例大小、显示位置、标题栏及按钮等一系列与窗口的显示效果相关的功能。
  - 键盘鼠标的消息投递负责将键盘鼠标发送来的消息发往到活动窗口，这个过程会涉及窗口管理器对活动窗口的仲裁。
  - 活动窗口仲裁会依据鼠标采用的仲裁模式（包括鼠标跟随式、鼠标按下式等）来确定正在操作的窗口。

1.3 编写操作系统需要的知识

鉴于操作系统是与硬件设备紧密相连的软件程序，所以操作系统的编写自然会涉及软件和硬件两个方面。

1. 硬件方面

首先，我们要根据硬件电路掌握处理器与外围设备的电路组成，通俗一点说，就是处理器和外围设备是怎么连接的。当掌握电路组成后，进而可以知道处理器如何控制外围设备，以及采用何种方式与它们通信。对于ARM这类片上系统而言，它们与外围设备的连接方法非常灵活，所以这部分内容必须要掌握；而通用的PC平台的连接方法相对固定得多。因此，编写一个PC平台的操作系统，对硬件电路的掌握要求会比较宽松。

其次，既然清楚了硬件电路的连接，下一步就是阅读这些硬件设备的芯片手册。芯片手册会详细描述芯片的硬件特性、通信方式、芯片内部的寄存器功能，以及控制寄存器的方法。不管何种硬件平台，硬件设备的芯片手册都同等重要，不了解这部分内容也就无法与硬件设备通信。一般情况下，操作系统开发人员会更关注处理器如何与这些硬件设备通信、如何控制它们的寄存器状态，而硬件工程师则会更关注芯片的工作环境、温度、工作电压、额定功率等硬件特性指标。

所以在硬件方面，掌握硬件电路、处理器和外围设备的芯片手册即可。其中，处理器芯片手册会介绍如何初始化处理器、如何切换处理器工作模式等一系列操作处理器的信息与方法，这些知识为操作系统运行提供技术指导。硬件芯片手册会对设备上的所有寄存器功能进行描述，我们根据这些寄存器功能方可编写出驱动程序。

2. 软件方面

至于软件方面，只要熟练运用汇编语言和C语言就足够编写操作系统了。

汇编语言主要用于控制和配置处理器，例如引导启动处理器、配置处理器运行状态、进程切换、中断和异常处理程序、设备I/O端口操作等必须操作寄存器的工作，或者是对性能要求极为苛刻的场景，这些工作C语言几乎无法实现。

C语言是编写操作系统的主要开发语言，它以简单、高效、使用灵活等特性深受底层开发人员的喜爱。而且，它在内嵌汇编语言以及与汇编语言的相互调用方面都表现得非常自然，只要遵循C语言的标准方法就能实现这些功能。

除了熟练使用开发语言之外，操作系统作为所有资源的管理者，一些灵活高效的算法也是必不可少的。从基础的链表结构，到树状结构，再到图状结构，操作系统会根据不同的应用场景有选择地使用它们。此外，一些灵活的编程技巧也必不可少。像内核异常处理程序的错误对照表，其原理是在程序容易出错的地方提前写出错误处理函数，并将出错地址和处理地址记录在错误对照表内。当错误发生时处理器会自动捕获错误地址，操作系统会从错误对照表里检索出对应的处理地址，并加以执行。这个过程必须借助链接脚本巧妙设计地址空间，才能组建错误处理对照表。

综上所述，编写一个操作系统必备的知识并不多，只需掌握汇编语言和C语言，能够看懂硬件电路图和硬件芯片手册即可。如果期望操作系统运行得又快又稳，那么还需要在兼顾空间开销和性能损耗的同时适当使用高效算法。所以，编写一个操作系统不难，难的是通过巧妙的方法让它运行得更高效、更人性化。