操作系统逻辑设计

The Logical Design of Operating Systems

〔美〕A·C肖著　任宏谋　杨晋兴等译　陈华瑛　刘熙明　校

操作系统逻辑设计

〔美〕A．C．肖　著
任宏谋　杨晋兴　等译
陈华瑛　刘熙明　校

人民邮电出版社

The Logical Design of Operating Systems

by Alan C. Shaw

Prentice-Hall, Inc.

1974

内 容 提 要

本书专门讲述操作系统的概念、原理和设计方法。书中深入讨论了进程、进程的同步和通信、多道程序、资源共享、系统核心、系统死锁等重要概念，详细介绍了进程和资源的控制、输入输出缓冲、主存管理、信息共享、文件管理等一系列技术，并提供了许多可供参考的设计实例和算法。

本书可供计算机软件工作者、大专院校有关专业师生参考。

操作系统逻辑设计

〔美〕A. C. 肖 著

任宏谋 杨晋兴等译

陈华瑛 刘熙明 校

人民邮电出版社出版

北京东长安街27号

河北省邮电印刷厂印刷

新华书店北京发行所发行

各地新华书店经售

开本：850×1168 1/32　　1984年1月第一版

印张：11 页数：176　　1984年1月河北第一次印刷

字数：288 千字　　印数：1—9,000 册

统一书号：15045·总2787—有5330

定价：1.70 元

译 者 的 话

本书是操作系统基本概念和基本原理方面--本较好的参考书。阅读此书需要计算机结构、程序设计语言(特别是算法语言 *ALGOL* 60) 等方面的基础知识。

书中详细阐述了操作系统的主要概念和原理，全面介绍了操作系统的三大组成部分——进程管理、资源管理、文件系统的功能及其设计方法，对诸如交互进程、临界区、进程和资源的控制、存储管理、输入输出缓冲、信息和代码的共享、死锁的检测及预防、系统故障恢复等一系列问题都作了相当深入的讨论。本书不专注于某一具体机器上操作系统的实现，而是提供许多实际系统中的例子，以说明原理的应用；还提供了一些有用的算法 (如第三章) 。

本书的主要特点是运用数据结构及在数据结构上的操作来说明一些重要的概念、原理和算法。特别是第七章，运用这种方法构造了一个系统核心，清晰地说明了一般系统中核心部分的主要功能。书中的算法大都是用*ALGOL*60描写的。第八章中采用图论方法对死锁问题进行了详细的讨论。

除署名译者外，冯耀霖、何永洛两同志参加了本书的部分翻译工作。六三一研究所原操作系统组、情报资料室及有关同志在本书翻译过程中曾给予大力支持，在此表示衷心感谢。

限于译者水平，译文中有不妥和错误之处，恳请读者批评指正。

序　　言

计算机操作系统是人类设计的许多最复杂的“系统”之一，我们能够了解并且有条不紊地组织这种复杂事物只不过是最近的事情。本书是一本操作系统原理教科书，其重点放在多道程序设计方面。我只打算介绍设计和了解这些系统所需的一些概念和技术，而不去详细讨论操作系统 x 怎样在机器 y 上实现；但是，为了解释清楚具体原理如何应用，书中也列举了许多从实际系统抽取出来的例子。我之所以选取“逻辑设计”这一书名，是为了强调我对操作系统各个部分的逻辑结构和相互影响，以及对思考这些问题的方法等方面的关心。

本书是为那些具有机器结构、汇编语言、程序设计语言和数据结构等基本知识的计算机科学大学生和专业人员编写的。阅读本书时必须预先具备的基础知识可从上述四门学科的第一学期课程中得到，这四门学科大约相当于计算机协会（ACM）在课程表68①中推荐的四门课程B_2、I_1、I_2和I_3。在写这本书时，就在科内尔大学和华盛顿大学把它用作研究生一学期课程的主要教本。本书无论作为大学研究生还是高年级学生的一学期或两学期课程都很合适，它包含了在课程表68①的课程I_4中和最近*COSINE*报告①中所提出的几乎所有的课题。

我认为，把操作系统这个题材划分成进程管理、资源管理和文

---

① ACM Curriculum Committee on Computer Science. *Curriculum* 68, 计算机科学的专科学校大纲介绍。*Comm ACM*, 11, 3 (March 1968), 151—197。

① COSINE Committee of the Commisson on Education. 操作系统原理的大学教程（主席是丹宁 Denning, P. J.）. Commission on Education National Academy of Engineering, Washington, D. C., 1971.

件系统这三个相关部分，是最适宜的。本书分九章，每章涉及到上述一个部分或几个部分的若干方面。第一章概述系统的硬件和软件的组织结构，其中包括发展概况和理论基础。在第二章中，我通过每次只处理一个作业的成批系统这一简单样本，来介绍连接装入程序和输入输出方法的基本思想。第三章建立了交互进程的一种模型，这一模型既被用为描述系统的工具，也用为解决进程通信和同步问题（包括第二章介绍的某些问题）的一种结构。第四章介绍多道程序系统：本章用前几章出现的材料为基础，讨论了多道程序设计对硬、软件提出的各种要求，用户和系统程序员所看到的“虚拟机”以及设计方法。第五章研究实存和虚存的管理技术。接下去的一章（第六章）继续研究主存资源，但着眼于实存和虚存系统中单副本的信息共享问题。在第七章中把进程和资源管理的思想统一起来：在这一章中叙述了一种综合核心，并将其作为一个模型来考察系统的数据结构、输入-输出进程、中断处理以及调度方法。第八章给出系统死锁的详细论述，并且介绍了既适用于顺序可重用资源（普通资源），也适用于消耗性资源（消息资源）的死锁检测、恢复和防止方法。最后一章（第九章）讨论文件系统的基本组成部分，其中一节专门介绍故障恢复。

本书包括有许多练习题，希望读者一定要做这些练习。在学习计算机系统的一些新思想时，特别重要的是给学生机会通过程序设计方案来应用这些思想。但有价值而容易处理的方案是不容易设计出来的；由于这个原因，我还增设了附录，它包括一个大型但却容易处理的多道程序设计方案的详细说明，这一方案我已成功地应用数次。

我力求给所有材料仔细地注明参考文献，这样既能使读者更深入地继续探讨某些领域或者得到另外的观点，同时又能把荣誉让给每种技术或思想的提出者。所有参考文献都汇集在书末。课文中引用参考文献时，先给出作者的姓，接着注明日期，例如Dijkstra，1965*b*。

ALAN C.SHAW

Seattle, Wash.

目　　录

第一章　计算机系统的组织

“逻辑设计”这一术语是计算机设计者用来描述以布尔代数为基础设计开关网络的系统设计方法。本书则是在更广泛的意义上使用这一术语来表明研究操作系统的一般方法，用这种方法来对操作系统进行体系设计以及研究它们的组织和性能。叙述重点是在一般原理而不在特别的“技巧”上；因此，既不仔细探讨编码技术，也不研究某个特定商用系统的情况，只是在说明某些观点时，引用了许多商用系统的例子。

本章考查计算机硬件和软件的历史发展过程，简单概述操作系统的组织和功能，以及从几种不同观点来讨论系统程序，通过这样三条途径引入这一课题。首先，定义几个基本术语。

1.1　几个定义

诸如“操作系统”、“分时”、“多道程序设计”一类术语，都还没有能为人们普遍接受的确切定义，在某些小型系统的理论研究文章中，也许有例外。这些术语只表示操作的组织、功能、行为或方法的某些类型。鉴于这一点，我们要对几个通常用来描述系统的重要术语给出非形式的定义。

操作（管理、监督、执行）**系统**（OS）是一组**有组织的**（系统）程序，在机器硬件和用户之间起着一种接口作用，它给用户提供了一组功能，用以简化程序的设计、编码、调试和维护；同时，它也控制资源的分配，以确保有效地操作。

有三类“纯”操作系统，我们从用户与其作业之间所允许的交互作用的类型及允许的系统响应时间来刻划它们。

1. **成批处理OS**　在这样的系统中，把用户作业分成若干批，顺序地提交给输入设备。在处理期间，用户与其作业无交互作用。用户同他的作业是完全隔开的，因而系统响应时间和作业解题周期相等。如果能估量出解题周期是在几分钟或几小时之内，则一般说来，已经令人满意了。因此，这种OS要采用比较灵活的调度策略。

2. **分时OS**　它给许多联机用户同时提供计算服务，允许每个用户与其计算交互作用。通过在几个用户中共享处理机时间和其它一些资源的办法，并以保证对每个用户命令在几秒钟内予以响应的方式工作，就可以达到同时访问的效果。给每个用户进程分配一个小的“时间片”（通常在毫秒级范围内）来使用计算机；如果一个进程用完其时间片时还未完毕，则将其中断，并放入等待队列，然后轮到另一进程使用机器。

3. **实时OS**　它服务于联机外部进程，这些进程在响应时间上有严格的约束。来自外部进程的中断信号请求系统注意，如果不迅速地（究竟应迅速到微秒、毫秒还是秒级的程度，由该进程决定）处理它们，则该外部进程将会严重地降低效能，甚至完全失去作用。往往针对某种特定的应用（比如过程控制）设计这样的系统。

一个特定的OS既可考虑提供成批处理、分时或实时作业中的某一种，也可以考虑三种功能都提供。例如，实时和分时的系统，在既无联机又无外部活动时，通常就在“后台”处理成批作业。

实现OS最常用的方法是利用多道程序设计。一个**多道（多道程序设计**的）OS（MS）是这样一种系统，它在主存中同时存入多个用户程序，这些活动着的用户作业共享处理机时间、存贮空间以及其它一些资源。这种资源的共享还延伸到操作系统之中；在大多数大型系统中，组成OS的程序本身也编入多道。

OS同时处理几个作业的另一个方法是通过**对换**实现的。对换的OS，在辅存中存放着若干作业，但任一时刻在主存只放一个作

业；系统转换到另一个作业的办法，是将当前作业移出主存，然后从辅存装入选上的作业。如果当前作业尚未完成，就在以后某个时刻再对换进来。这种技术主要用于小型分时系统。

最后一个定义，**多重处理**，描述的是一个系统的硬件配置情况，有时易和多道程序设计的概念相混淆。多重处理的计算机系统，就是有多台独立处理机的计算机硬件的组合体。独立处理机包括中央处理机（CPU）、输入输出（IO）处理机、数据通道以及专用处理机（比如运算器）。但是这个术语最常见的是指多台中央处理机（CPU）。

本书主要讨论多道程序操作系统，——讨论与其它的系统相比较之下，这种组织的优缺点，还要讨论共享时间、空间以及共享其它资源的要求及技术。

1.2 算法的表达工具

程序设计语言Algol 60 (Naur, 1963) 及其公认的变型，将成为我们说明算法的主要工具。其所以选用 Algol 而不选英语、框图、汇编语言或其它高级语言，有下述几个原因：

1. 在发表的文献中，Algol 的语法和语义都有明确定义，并没有二义性。

2. 多年来，Algol成功地用作国际出版算法语言，

3. 必要时，Algol 型的描述能够足够“高级”地避免许多内务细节问题。反之，用其“低级”方式又能直接变换成机器语言①。

4. 作者及其许多学生和同事都认为：在推导、组织和分析算法时，Algol是一种强有力的表达工具〔有关 Algol的介绍和原报告

①虽然我们也遵从结构程序设计的一般原理(例如，见Dijkstra1969,SIGPLAN1972)，但是读者仍然会在我们的某些程序中找到 *go to*语句。*go to*并未统统消除掉，因为它有利于另一些结构。当用明显的方法描述机器一级的活动以及明显地展示控制流程时，*go to*语句很有用。因此，在按严格的方式使用*go to* 语句时，也可以产生结构良好的程序。

文本，见Rosen（1967，*pp*.48-117）〕。

Algol过程的例子

在操作系统中，常常使用树形的数据结构，例如表示进程的层次结构（第七章），表示文件目录（第九章）等。

二叉树由结点的有穷集构成，树可是空的，也可分成一个根结点和两个互相分离的二叉树，即左、右子树（Knuth，1968）。假定用三元组（*Data*〔*n*〕，*Left*〔*n*〕，*Right*〔*n*〕）表示二叉树中每个结点*n*，其中*Data*〔*n*〕为正整数，*Left*〔*n*〕和*Right*〔*n*〕分别表示指向左、右子树之根的非负指针。对空树来说，约定结点指针$n=0$。对每个结点*n*，我们假定：

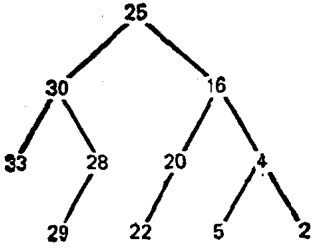
$$(1)\quad \begin{array}{ll} Data[n]<Data[x] & \text{对所有的} x \in Leftsubtree(n) \\ Data[n]>Data[x] & \text{对所有的} x \in Rightsubtree(n). \end{array}$$

图1—1给出了这种树的一个例子〔有时按这种方式组织符号表，以便可以很容易地进行扩充和查找（*Gries*，1971）〕。下面给出一个Algol过程*Treesearch*（*root*，*arg*，*m*），它在根为*root*的树中查找节点*n*，使*Data*〔*n*〕=*arg*；如果查到，则返回值为true，并将*m*置为所找到的结点；否则就返回值为false。此算法直接使用二叉树的递归定义：

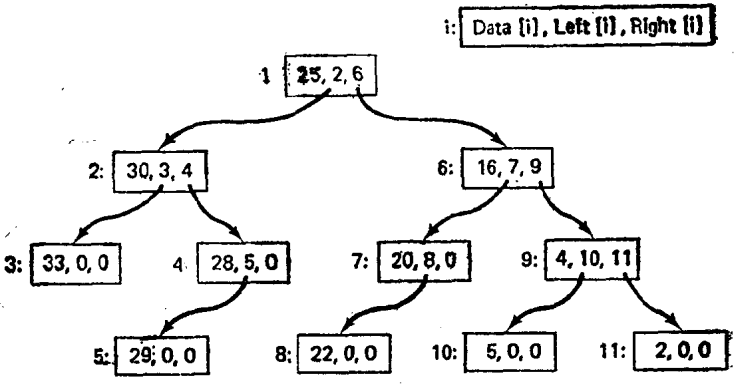
```
Boolean procedure Treesearch(root,arg,m);
value root,arg; integer root,arg,m;
comment Data[],Left[],Right[]are assumed global
to this procedure. Search the tree with root root for
the node n such that Data[n]=arg;
```

〔注解：假定Data〔〕，*Left*〔〕，*Right*〔〕是过程的全程量。从根为*root*的树中找出结点*n*，使*Data*〔*n*〕=*arg*

```
if root=0 then Treesearch:=false else
```



(a)二叉树



(b) 内部表示

图 1-1 为快速分类、查找和插入而组织的二叉树

```
begin
  integer d;
  d:=Data[root];
  if arg=d then begin m:=root; Treesearch:=true end
  else
  if arg>d then Treesearch:=Treesearch(Left[root],arg,m)
  else Treesearch:=Treesearch(Right[root],arg,m)
end
```

练　　习

写出一个*Algol*过程*Addtotree*(*root*, *n*),其中的树是根为*root*、结点次序如(1)规定的二叉树，该过程把一个结点*n*加入此树，使其仍保持(1)的次序。再写一个过程，按递增顺序打印树的数据；可利用任一方便的原语，比如*Write*(*x*)，作为打印变量*x*的输出调用。

1.3　历史发展概况

本节简短描述一下计算机硬件和软件系统的历史发展过程。更详细的讨论和书目提要可在S.Rosen(1969)和R.Rosin(1969)中找到。

1.3.1　早期的系统

第一台存有程序的数字计算机真正开始执行指令的时候，大约是1949年。从那时起到1956年止的这段时间内，计算机操作的基本结构和方式保持相对的稳定（有一些有远见的尝试，但大部分未成功）。经典的von Neumann结构是以严格顺序地执行指令（包括IO操作在内）为基础的。每当装入和运行程序时　用户就要在控制台上直接同机器进行联机工作，比如，置寄存器、一步步执行指令、查看存贮单元等，以最低的机器一级的方式与其计算进行通常的交互作用（分时系统吸收了这种操作方式的优点，但它是在较“裸机”更高一级上提供这种操作的）。程序是用绝对机器语言（十进制或八进制表示）书写的，而且还要在前面放上一个绝对装入程序。

回顾一下绝对装入过程是有益的，因为甚至到目前　它们仍代表了裸机上任何软件系统的起点。任何计算机均有装入按钮之类的