Linux 操作系统及应用
附录 D — 操作系统基本知识

唐晓晟　李亦农
txs@bupt.edu.cn　hoplee@bupt.edu.cn

Beijing University of Posts and Telecommunications (BUPT)
School of Information and Communication Engineering

内容简介

知识铺垫 I

- 计算机软件分为系统软件和应用软件
  - 系统软件，用于管理计算机本身及应用程序
  - 应用软件，实现用户所需要的功能
- 操作系统 (Operating System) 是最基本的系统软件，控制计算机的所有资源并提供应用程序开发的基础。
- 现代计算机系统包括：若干处理器、主存储器、时钟、终端、磁盘、网络接口和其他输入/输出设备。
- 操作系统的出现，避免了程序员直接面对这些复杂的硬件控制问题，程序员可通过其提供的接口或虚拟机 (Virtual Machine) 来进行程序设计，间接操控各种硬件资源
- 计算机系统典型层次结构如下表所示：（这个表中，下面三层为硬件，最上面一层为应用程序，其他两层为系统程序）

知识铺垫 II

数据库系统	银行系统	文字处理
编译器	编辑器	命令解释器
操作系统		
机器语言		
微程序		
物理设备		

物理设备 集成电路、芯片、连线、电源、阴极射线管和相关物理装置

微程序 microprogram，通常存于只读存储器中，实际上为一个解释器，解释类似 ADD、MOVE、JUMP 等机器指令并执行

机器语言 一般包括 50 到 300 条指令，通常从事数据传送、算术运算和数值比较等操作

知识铺垫 III

操作系统[1] 隐藏上述所有复杂性，为程序员提供更加方便的一套指令，操作系统在核心态 (kernel mode) 或称管态 (supervisor mode) 下运行，受硬件保护，避免遭受到用户的破坏

应用软件 用户编写，解决诸如商业数据处理、工程计算或者电子游戏等特定的问题，一般在用户态 (user mode) 下运行

[1]解释器 (shell)、编译器、编辑器等类似的应用程序本身并不算是操作系统的组成部分，通常由计算机厂商提供

概念及历史

- 本节内容主要介绍计算机操作系统的“前世今生”

什么是操作系统

作为扩展机器的操作系统 完成将硬件细节（如具体磁盘操作）与程序员的隔离，提供方便、间接的使用方式，相当于为用户提供一台等价的扩展机器，或称虚拟机 (extended machine)，此为自顶向下看的观点

作为资源管理器的操作系统 自底向上看，操作系统用来管理一个复杂系统的各个部分，包括处理器、存储器、时钟、磁盘、终端、磁带设备、网络接口、打印机等，操作系统的任务是在相互竞争的程序之间有序地控制对各种资源的使用，协调各种请求冲突

操作系统历史 I

按照计算机系统结构划分，第一台真正的数字计算机由英国科学家 Charles Babbage(1792-1871) 设计，由于纯机械式，精度无法满足要求，没有能够成功运行，当然，无操作系统

操作系统历史 II

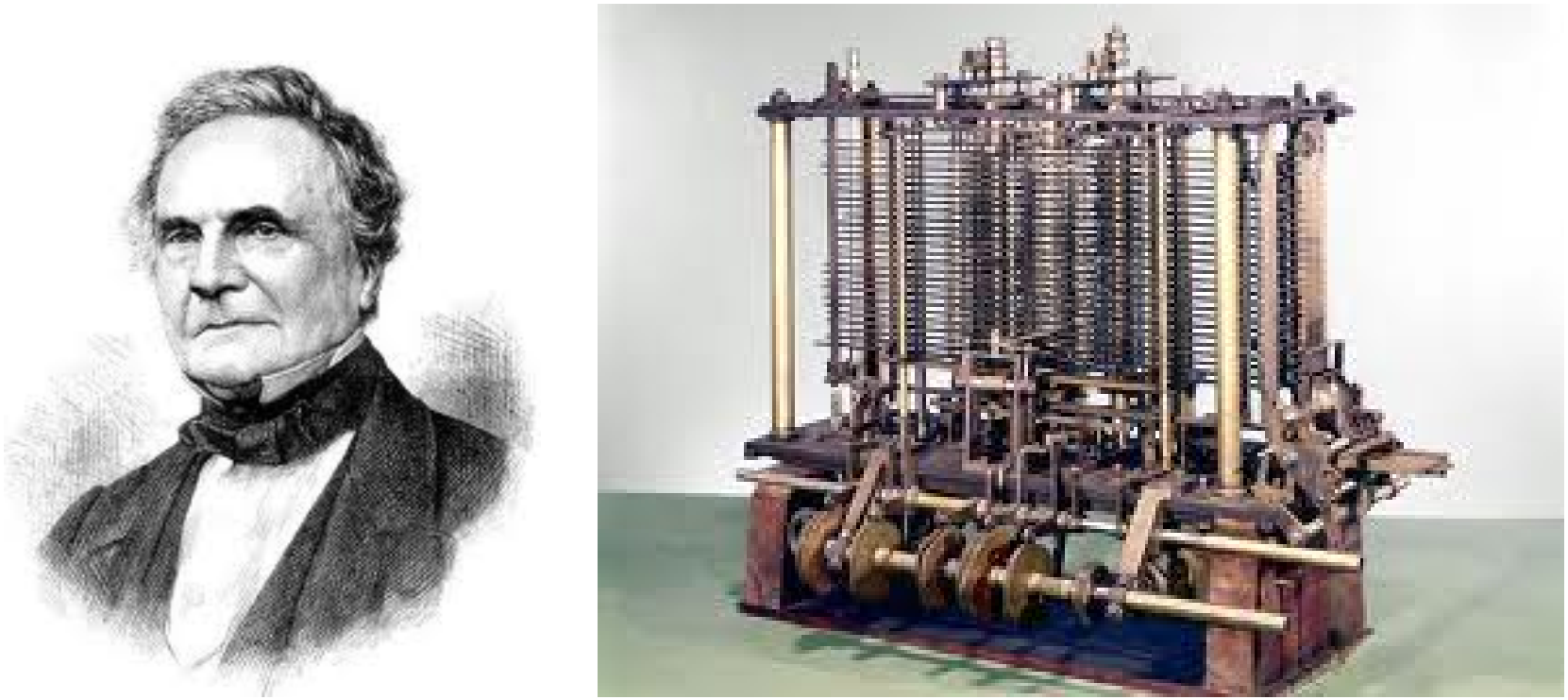

Figure: Charles Babbage and his Analytical Engine

操作系统历史 III

- 第一代计算机 (1945-1955)：真空管和插件板，数万真空管组成，占据几个房间，由专门小组来设计、制造、编程、操作和维护，采用机器码编程，通过插板上的硬连线来控制，无程序设计语言，无操作系统，主要针对数值计算，后期出现了穿孔卡片替代插件板，其他过程依然如旧

操作系统历史 IV

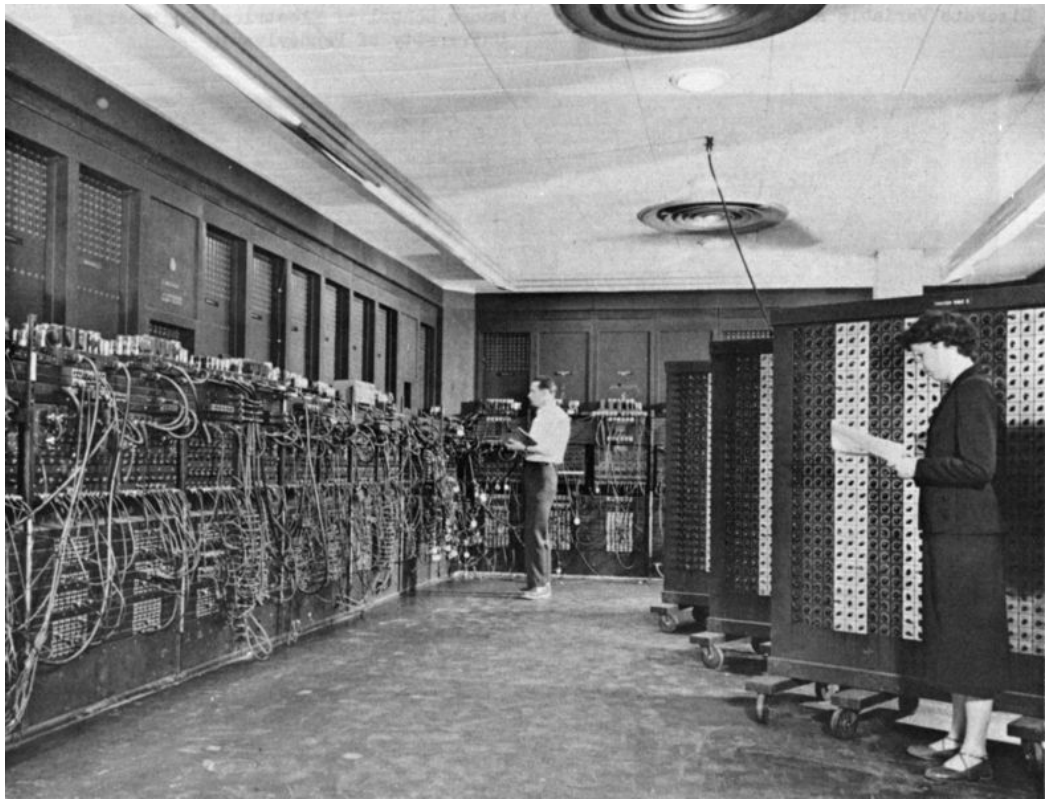

Figure: 第一代计算机（ENIAC）

操作系统历史 V

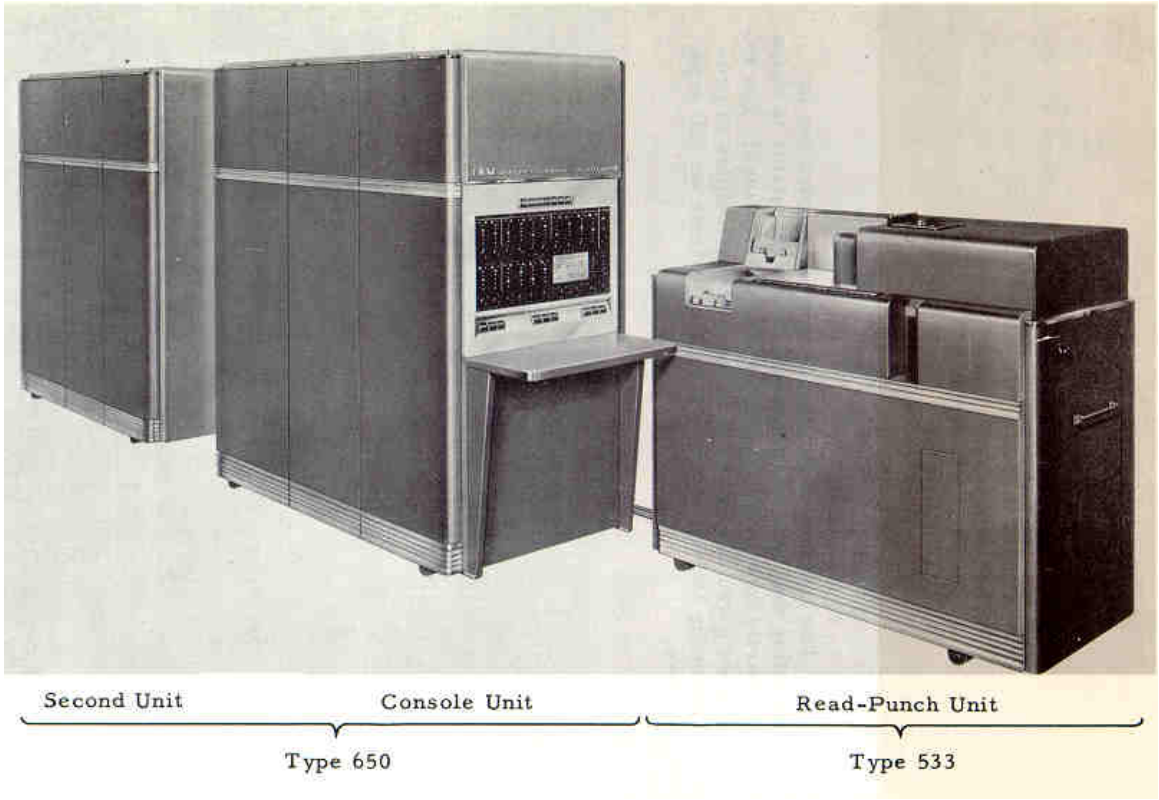

Figure: 第一代计算机（IBM650）

操作系统历史 VI

- 第二代计算机 (1955-1965)：晶体管和批处理系统，计算机已经很可靠，可批量生产，设计、操作、程序和维护人员第一次有了明确的分工，出现了 job 的概念（一个或一组程序），由卡片承载，专门操作员负责运行，为减少机时浪费，通常采用批处理系统 (batch system)，输入室收集全部作业，用便宜的计算机将其读到磁带上，然后用昂贵的计算机完成真正的计算（过程中，从磁带上读取多个作业的特殊程序即为操作系统的前身），计算结果输出到磁带上，拿到另外的机器上进行脱机（off line，即不与主计算机联机）打印。典型的作业结构如图所示，此时计算机主要用于科学与工程计算，大多采用 FORTRAN 和汇编语言编写，典型的操作系统为 FMS(FORTRAN Monitor System)（图8）和 IBSYS(IBM 为 7094 机配备)

操作系统历史 VII

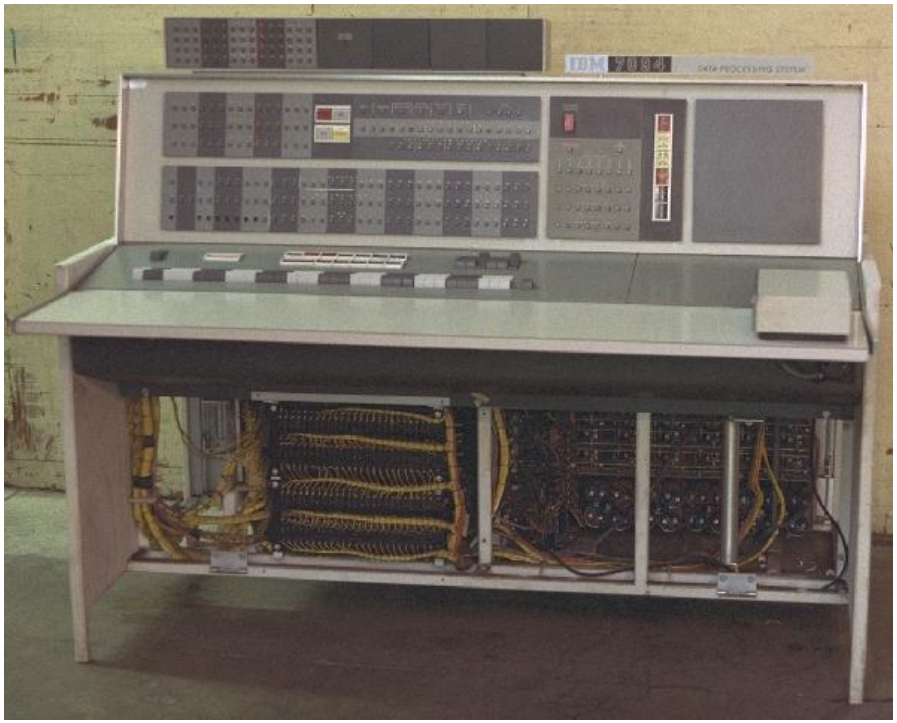

Figure: 第二代计算机（IBM7094 Main Console）

操作系统历史 VIII

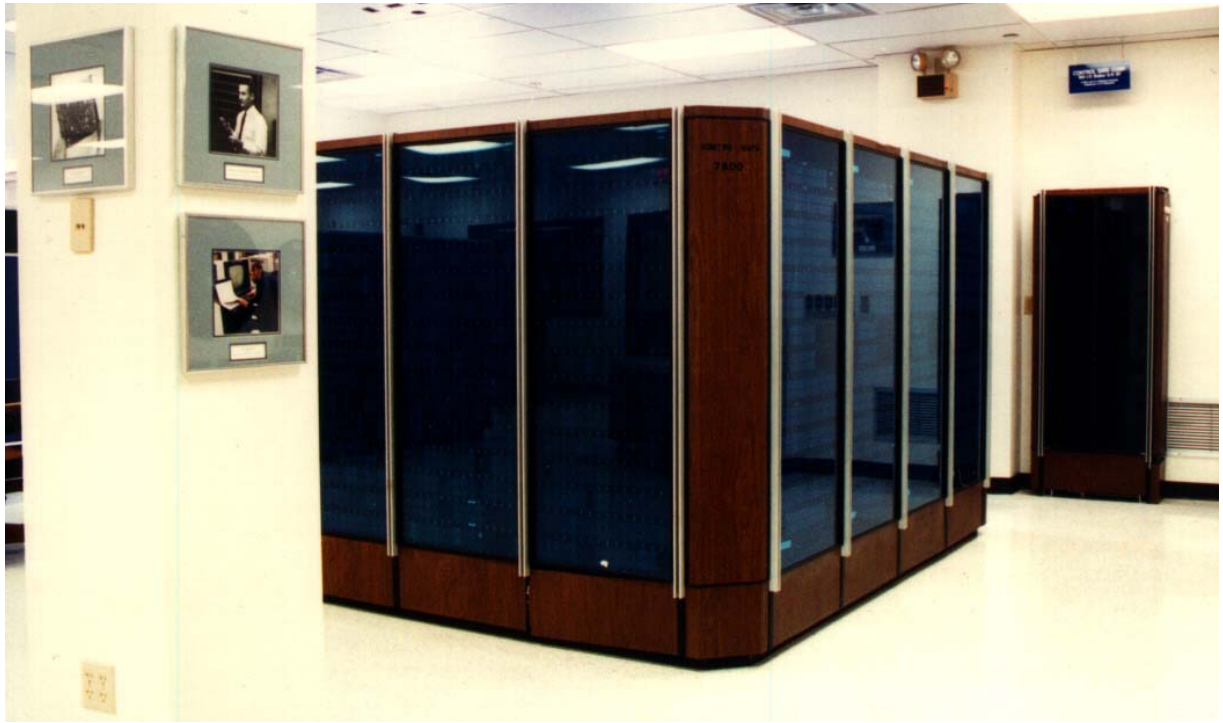

Figure: 第二代计算机（CDC7600）

操作系统历史 IX

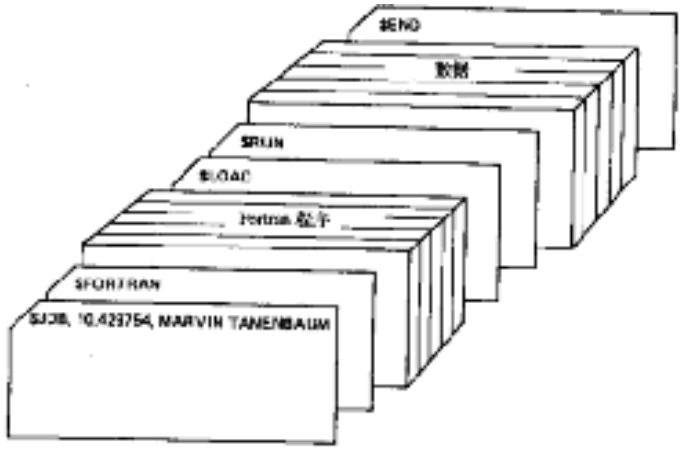

Figure: FMS

操作系统历史 X

- 第三代计算机 (1965-1980)：集成电路芯片和多道程序，集成电路出现，性价比、稳定性得到大规模提升，IBM OS/360 的出现，推动了操作系统发展，主要关键技术包括，多道程序设计 (multi-programming) 的引入来避免 CPU 时间浪费，splooling(Simultaneous Peripheral Operation On Line，同时的外部设备联机技术，也叫假脱机技术)，加快作业从磁盘读出并装入内存运行。但是，将程序作业化，会导致程序员们长时间的等待，这导致了分时系统 (timesharing)[2]的出现，每个用户一个联机终端，分时使用主机 CPU 资源，后 MIT、贝尔实验室、美国通用电气决定共同开发 MULTICS(MULTiplexed Information and Computing Service)[3]，目标是满足整个波士顿用户的计算需求，该系统最终由 MIT 研制成功

操作系统历史 XI

Figure: 第三代计算机

操作系统历史 XII

Figure: 第三代计算机（IBM360）

操作系统历史 XIII

- 第四代计算机 (1980-1990)：个人计算机、大规模集成电路的发展，推动了个人计算机时代的到来。其特点是：价格便宜及软件界面友好。计算资源的普及，也使得为个人计算机编制软件成为一项重要的产业。主流操作系统有：微软的 MS-DOS 以及 UNIX。随着 Internet 的发展，网络操作系统 (Network Operating Systems)[4]和分布式操作系统 (Distributed Operating Systems)[5]应运崛起。

操作系统历史 XIV

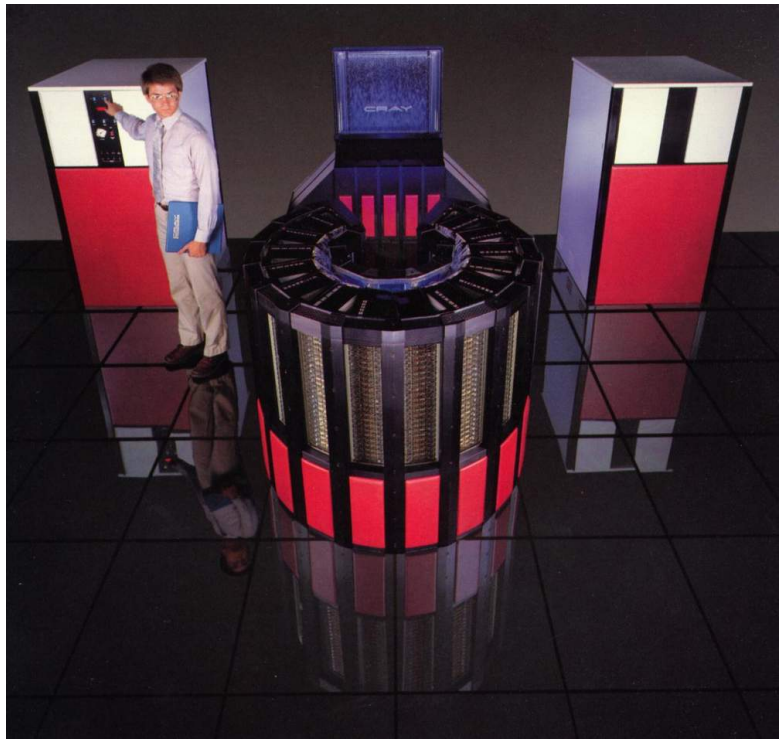

Figure: 第四代计算机（PC and Cray-2）

[2]第一个分时系统 CTSS(Compatible Time-Sharing System)，1962 年，麻省理工学院

[3]后来，贝尔实验室 MULTICS 研发参与者 Ken Thompson，在一台无人使用的 PDP-7 上开发了一个简化的、单用户版的 MULTICS，导致了 UNIX 的诞生

[4]用户知道多台计算机的存在，用户能够远程登陆和操作，每台计算机运行自己本地的操作系统，有自己的本地用户

基本概念

- 操作系统与用户程序的界面由操作系统提供的“扩展指令”集定义，这些扩展指令传统上称为系统调用 (system call)
- 系统调用创建、删除和使用由操作系统管理的各种软件对象，其中最重要的是进程和文件
- 这些概念在 UNIX 和 MS-DOS 上的原理都是类似的，只是一些具体实现细节不同

进程 (process)

- 基本上是一个执行的程序，包括可执行的程序、程序数据与栈、有关程序计数器、栈指针、其他寄存器以及所有运行程序需要的信息
- 分时系统周期性地挂起一个进程，然后运行另一个进程
- 进程在挂起时，其所有信息都要保存下来，例如进程打开的若干个文件的准确位置等，这些信息保存在进程表 (process table)，其为一个数组或链表，每个进程占用一项 (entry)
- （挂起的）进程包括：地址空间（往往称为磁芯映像，core image）、对应的进程表项等
- 进程创建的进程叫做子进程 (child process)，进而得到进程树
- 进程之间可进行信息传送，也即进程间通信 (IPC)，信号机制是一种常见的方式
- 进程的拥有者信息非常重要，操作系统一般通过用户标识符来保存此类信息 (uid, gid)

文件 (file) I

- 文件相关的系统调用主要目的是隐藏磁盘和其他 I/O 设备的细节，提供简洁方便的接口来创建、删除和读写文件
- 大多数的操作系统支持将一组文件形成目录的形式来存放文件，目录项可以是文件或目录，这样就产生了层次文件系统
- 目录层结构中的每个文件都可以从根目录 (root directory) 开始的路径名 (path name) 来确定，绝对路径名包含了从根目录到该文件的所有目录清单，之间以斜线分隔，如/Faculty/Prof/Courses
- 任一进程都有一当前工作目录 (working directory)，相对路径的搜索从当前目录开始，该目录可通过系统调用改变
- 操作系统为文件提供保护手段，不同系统有不同的机制。访问文件前，会进行权限检查，若权限许可，系统将返回一个小的整数，称作文件描述符 (file descriptor) 或文件句柄 (handle)，供后续操作使用

文件 (file) II

- 多数操作系统提供一种抽象，以允许用户在不了解硬件细节的情形下完成 I/O 操作，这种抽象把 I/O 设备视作特殊文件 (special file)，让用户可使用与访问普通文件相同的系统调用来进行操作。共有两种特殊文件：块特殊文件 (block special file，如磁盘) 和字符特殊文件 (character special file，如终端、行式打印机、网络接口等)
- UNIX 或 MS-DOS 中，进程开始后，文件标识符 0 称为标准输入 (standard input)，文件标识符 1 称为标准输出 (standard output)，文件标识符 2 称为标准错误输出 (standard error)
- 管道 (pipe) 是一种连接两个进程的虚拟文件，实际应用时作为源进程的输出文件和目标进程的输入文件

系统调用 (system call) I

- 系统调用 (system call) 是用户请求操作系统服务的方式，用户通过调用对应的库过程来进行（如 read 函数）。在这一过程中，系统调用参数放到指定位置，如寄存器，然后发出 TRAP 指令（一种被保护的系统调用）给操作系统
- 库过程的目的是隐藏 TRAP 指令的细节，并使得系统调用更像普通的过程调用
- TRAP 之后，操作系统取得控制，会考察参数是否有效，如果是，完成请求，在某个寄存器中给出状态代码，通知是成功或失败，然后执行 RETURN FROM TRAP，将控制权返给库过程，库过程按照通常方式返回到调用方，把状态代码作为函数值返回（执行成功时一般返回 0，错误时一般返回某个负整数）

系统调用 (system call) II

- UNIX 和 MS-DOS 中的 READ 系统调用，需要三个参数：要读取的文件、存放数据的缓冲区和读多少个字节，C 语言中对 READ 的调用为`count = read(file, buffer, nbytes);`
- 可以认为，系统调用通常会创建进程、管理存储器、读写文件和从事诸如从终端读以及在打印机上输出等 I/O 工作

外壳 (shell)

- 操作系统是指从事系统调用的代码，编辑器、编译器、汇编程序、链接程序以及命令解释器虽重要，但不是操作系统的组成部分
- 外壳 (shell) 即为 UNIX 下的命令解释器，体现了许多操作系统的特性，很好地说明了系统调用的具体用法，同时也是终端用户与操作系统之间的界面
- shell 环境下，用户可以输入指令，命令 shell 创建进程完成任务，也可以使用重定向、管道等功能

操作系统结构

- 在了解了操作系统的外部特性（即程序员接口）的前提下，现在开始观察其内部的组成结构
- 我们通过考察四种已经尝试过的操作系统结构来获得一些感性认识

整体式系统 I

- 最常用的组织方式，常被形容为“大杂烩”，其结构实际就是“无结构”，整个操作系统为一堆过程的集合，每个过程都可以任意调用其他过程
- 此类系统，没有任何信息隐藏，每个过程都对每一个其他过程可见（与之对应的是分成那个若干个模块的系统，信息隐藏在模块内部，只允许从预先确定好的外部调用点访问这些模块）
- 操作系统提供的服务（系统调用）的调用过程
  - 参数装入预订的寄存器或堆栈，执行一条特殊的终端指令 (supervisor call 或 kernel call)，该指令将机器由用户态切换到核心态，将控制转到操作系统
  - 操作系统检查所调用的参数以确定应执行哪条系统调用
  - 调用相应的服务过程
  - 系统调用结束后，将控制返回给用户程序
- 其三层模型如图30所示

整体式系统 II

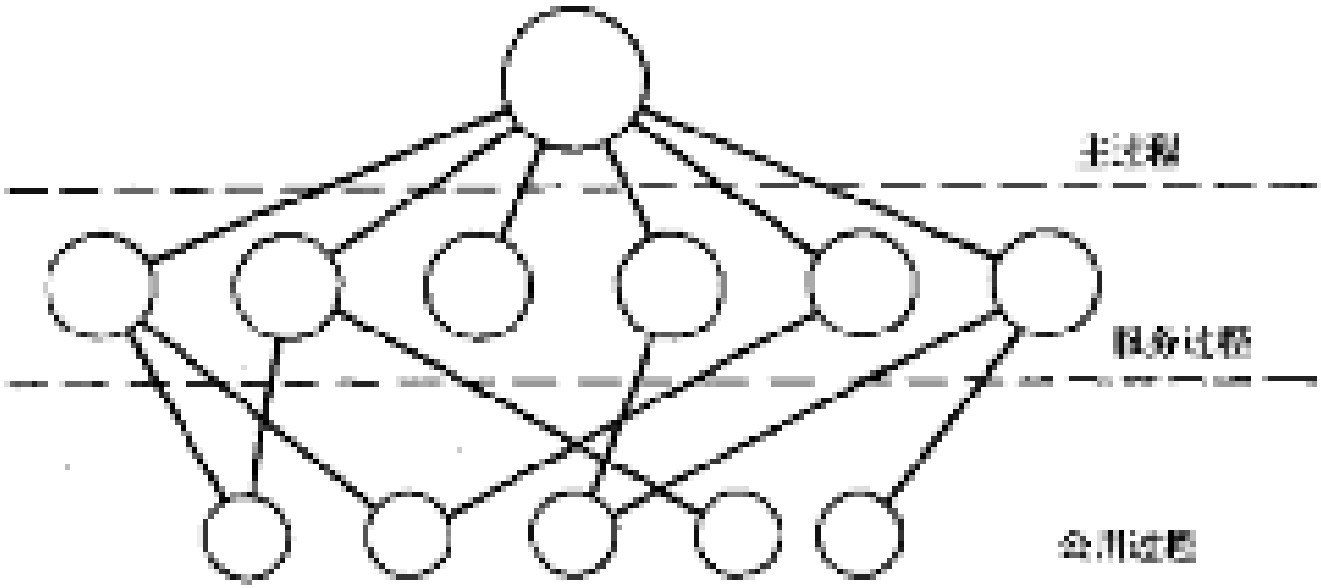

Figure: 简单的整体式操作系统结构模型

层次性系统 I

- 将图30进一步通用化，形成层次性系统，E.W.Dijkstra 在荷兰 Eimdhoven 技术学院 1968 年开发的 THE 系统，是按此模型构造的第一个操作系统（内存 32K 个字，每字 27 比特），每层功能说明如下（图32）

Figure: THE 操作系统结构

第 0 层 处理中断发生或定时器超时时的进程切换

层次性系统 II

第 1 层 分配进程的内存空间，支持交换空间的概念（磁鼓上 512K）

第 2 层 处理进程与管理员控制台之间的通信。可认为，每个进程都有自己的管理员控制台

第 3 层 管理 I/O 设备和相关的信息流缓冲区。这里，每个进程与抽象 I/O 设备交互，不必考虑设备的物理细节

第 4 层 用户程序不必考虑进程、内存、控制台或 I/O 设备等细节

第 5 层 系统管理员进程

- MULTICS 进一步拓展了这个概念。整个 MULTICS 系统由许多同心环 (ring) 构造成，内层环比外层环有更高的级别。当外环的过程调用内环的过程时，必须执行一条等价于系统调用的 TRAP 指令，同时进行严格的参数合法性检查。此方案由硬件实现，可保证对单个过程（内存中的一个段）的读、写和执行进行保护

虚拟机 I

- 最初由 IBM 研究中心开发完成 CP/CMS(Control Program/ Cambridge Monitor System)，后改名为 VM/370，其系统核心被称作虚拟机监控程序 (virtual machine monitor)，在裸机上运行，几乎具备了多道程序功能，向上层提供了若干台虚拟机
- 不同于其他操作系统的是，这些虚拟机不是那种具有文件等优良特征的扩展计算机，而仅仅是精确复制的裸机硬件，包括：核心态/用户态、I/O 功能、终端等真实硬件具有的全部内容（图34）

虚拟机 II

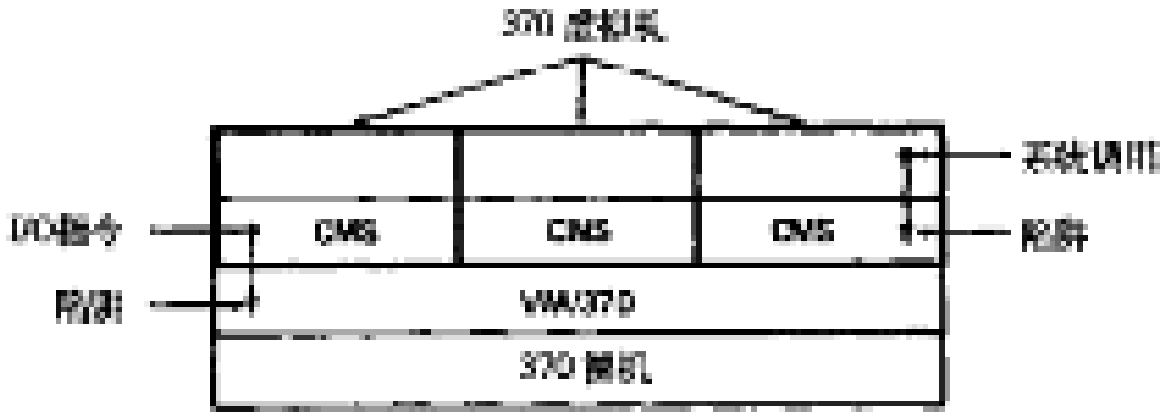

Figure: 带 CMS 的 VM/370 结构

- 每台虚拟机可以运行任何类型的操作系统，实际上也是如此，这个系统后来改名为会话监控系统，CMS(Conversational Monitor System)

虚拟机 III

- CMS 的程序在进行系统调用时，系统调用陷入其虚拟机中的操作系统，而不是调用 VM/370，就像在真正的计算机上一样，然后 CMS 发出硬件 I/O 指令，在虚拟磁盘上读写或执行各种操作，这些指令被 VM/370 捕获，作为对真实硬件模拟的一部分，最终真正完成操作
- 这样将多道程序的功能和提供扩展机器的功能完全分开后，它们各自都更简单、更灵活和易于维护

客户机/服务器系统 I

- 现代操作系统的发展趋势是，进一步发展这种将代码移到更高层次的思想，尽可能多地从操作系统中去掉东西，只留下一个很小的内核 (kernel)
- 通常采用的方法是，由客户机进程来实现大多数操作系统的功能，在需要某项服务（比如说读文件）时，客户机进程 (client process) 把请求发给服务器进程 (server process)，然后服务器进程完成此操作，并返回应答信息，其模型如图37所示

客户机/服务器系统 II

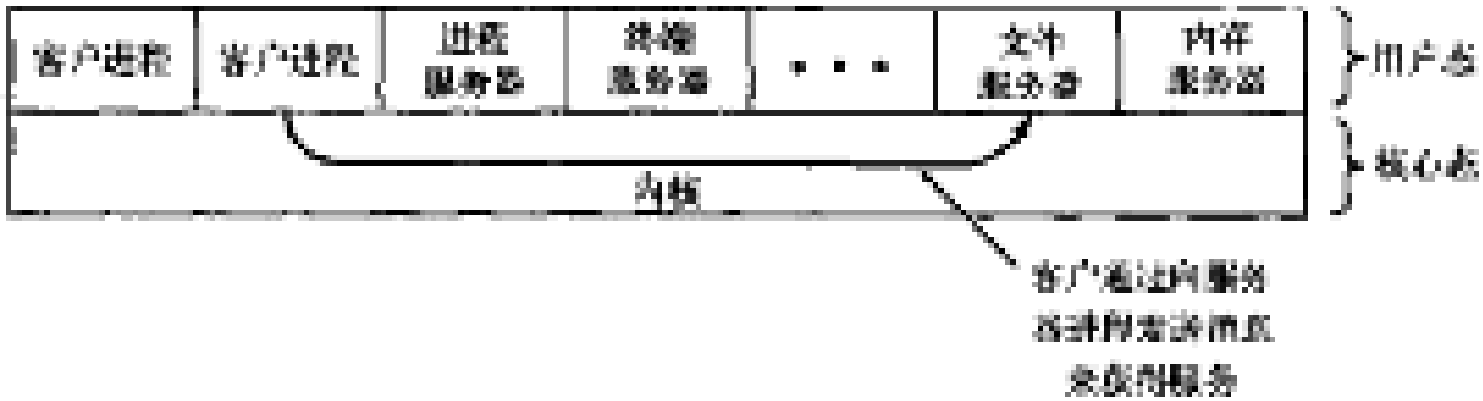

Figure: 客户机/服务器模型

- 操作系统内核的全部工作是处理客户机与服务器之间的通信。每个模块仅仅处理一个方面的功能，如文件服务、进程服务、终端服务或存储器服务等，易于管理。而且，服务器程序在用户态下运行，不直接访问硬件，出问题也不会导致整个系统崩溃

客户机/服务器系统 III

- 该模型也适用于分布式系统。如果客户机通过消息传递与服务器通信，那么客户机不需要知道这条消息是在本地处理的还是通过网络送给了远端主机上的服务器。两种情况下，客户机的处理都相同：发送请求、获取应答
- 对于一些仅在用户态进程很难完成的系统功能，有两种常用的解决方法：建立核心态的用户服务进程（如 I/O 设备驱动程序）；内核建立最小的机制（甚至不进行合法性检查），而把策略 (policy) 留给用户空间的服务进程（需要授权机制加以限制）

小结

- 操作系统从 50 年代开始发展，已经从只控制单台计算机发展到运行在以网络链接的大批机器上
- 后续的操作系统，称为网络式或分布式操作系统，但老式操作系统没有一个通用名称，如集中式 (centralized)、单处理器 (single-processor)、单 CPU(single-CPU) 或甚至称作为传统 (traditional) 操作系统，意思都是指只管理自身所在的、单台计算机的操作系统
- 还有一种多处理器操作系统 (multiprocessor operating system)，控制严密集成的包含二个或者更多的 CPU 系统，处于单处理器系统和分布式系统之间，兼有两种系统的特征
- 典型的操作系统包括四大功能模块：进程管理、存储管理、文件管理和设备管理

The End

The End of Appendix D: OS Fundamental.