现代操作系统(第一章)

含义

- 作为扩展机器的操作系统(自顶向下)
  - 原始结构
    - 体系结构
      - 指令集
      - 存储组织
      - I/O
      - 总线结构
    - 特点：编程困难，不易理解
  - 抽象处理
    - 举例：OS对底层SATA硬盘进行抽象
      - 硬盘驱动
        - 文件
        - 抽象处理
      - 抽象处理
    - 操作系统包含很多用于控制输入/输出设备的驱动
  - 抽象是管理复杂性的一个关键，好的抽象可以把一个几乎不可能管理的任务划分成两个可管理的部分
    - 第一部分：有关抽象的定义和实现
    - 第二部分：随时使用这些抽象解决问题
  - 抽象是理解操作系统的关键
  - 操作系统的一个主要任务是隐藏硬件，呈现给程序(以及程序员)良好、清晰、优雅、一致的抽象
- 作为资源管理者的操作系统(自底向上)
  - 操作系统的任务是在相互竞争的程序之间有序地控制对处理器、存储器以及其他 I/O 接口设备的分配
  - 操作系统的主要任务是记录哪个程序在使用什么资源，对资源请求进行分配，评估使用代价，并且为不同的程序和用户调节互相冲突的资源请求
  - 资源管理方式：多路复用(共用)资源
    - 时间上复用
      - 当一种资源在时间上复用时，不用的程序或用户轮流使用它。先第一个获得资源的使用权，然后下一个，以此类推。
        - 单CPU时间片轮训
        - 单个打印机的共享
    - 空间上复用
      - 每个客户得到资源的一部分，从而取代了客户排队
        - 内存复用：通常若干个运行的程序之间分割内存，这样每一个运行程序都可同时入驻内存(提高了整体效率)，但是引起的公平和保护问题由操作系统解决
        - 磁盘复用：一个磁盘同时为许多用户保存文件。分配磁盘空间并记录谁正在使用哪个磁盘块，是操作系统的典型任务

操纵系统结构

- 操作系统与运行操作系统的计算机硬件联系密切，操作系统扩展了计算机指令集并管理计算机的资源。
- 从概念上将，一台简单的个人计算机由CPU、内存以及I/O设备都由一条系统总线连接起来并通过总线与其他设备通信。(现代个人计算机结构更加复杂，包括多重总线等)
- 处理器CPU
  - 执行流程：每个CPU基本周期中，首先从内存中取出指令，解码以确定其类型和操作数，接着执行指令，然后取指、解码并执行下一条指令
  - 专门指令集：每个CPU都有一套可执行的专门指令集。x86处理器不能执行ARM程序，ARM程序不能执行x86程序
  - 寄存器：为了解决访问内存和2CPU执行指令时间不对等问题
    - 通用寄存器：保存关键变量和临时数据的寄存器
    - 专用寄存器：对程序员可见
      - 程序计数器：保存了将要取出的下一条指令的内存地址。指令取出后，程序计数器就被更新以便指向后随的指令
      - 堆栈指针：指向内存中当前栈的顶端，该栈保存了每个执行过程的栈帧。一个过程的栈帧中保存了有关的输入参数、局部变量以及没有保存在寄存中的临时变量
      - 程序状态字寄存器：包含了条件码位、CPU优先级、模式(用户态或内存态)以及各种其他控制位
  - 操作系统必须知晓所有的寄存器。在时间多路复用CPU中，操作系统经常会终止正在运行的某个程序并启动(或再启动)另一个程序，每次停止一个运行着的程序时，操作系统必须保持所有的寄存器值，这样在稍后该程序被再次运行时，可以把这些寄存器重新装入
  - 流水线：为了改善性能，CPU设计师放弃了同时读取、解码和执行一条指令的简单模型。许多现代CPU具有同时取出多条执行的机制。将读取、解码和执行分为了不同的单元。执行第n条执行时，可以对指令n+1解码，并且读取执行n+2。这种机制称为流水线
  - 超标量CPU：这种设计中，有多个执行单元。例如，一个CPU用于整数算数运算，一个CPU用于浮点算数运行，一个CPU用于布尔算数运算。两个或更多的执行被同时取出、解码并装入暂存缓存区。
  - 除了用在嵌入式系统中的非常简单的CPU之外，多数CPU都有两种模式，内核态和用户态(PSW中有一个二进制位控制这两种模式)
    - 内核态：CPU可以执行指令集中的每一条指令，并且使用硬件的每种功能。在台式机和服务器，操作系统在内核态运行。在大多数嵌入式系统中，一部分操作系统运行在内核态，其余部分则运行在用户态
    - 用户态：用户程序运行在用户态，仅允许执行整个指令集的一个子集和访问所有功能的一个子集。一般而言，在用户态有关I/O和内存保护的所有指令是禁止的
  - 系统调用：为了从操作系统中获得服务，用户程序必须使用系统调用以陷入内核并调用操作系统。TRAP指令把用户态切换成内核态，并启用操作系统。当有关工作完成之后，在系统调用后面的指令把控制权返回给用户程序。
  - 陷阱：计算机使用陷阱而不是一条指令来执行系统调用。其他的多数陷阱是由硬件引起的，用于警告有异常情况发生，如试图被零除或浮点下溢等。在所有情况下，操作系统都得到控制权并决定如何处理异常情况。如：停止程序或如程序提前设定了要处理某类条件，则将控制权返回给程序，让其进行处理。
- 存储器
  - 由于不存在理想状态下的存储器，极为迅速，充分大，非常便宜。所以存储器系统采用一种分层次的结构。
  - 寄存器: 典型访问时间 1(ns)，典型容量<1KB
    - 位于CPU中，和CPU材料相同
    - 典型的存储容量，32位CPU中为32x32位，64位CPU中为64x64位
    - 程序必须在软件中自行管理这些寄存器
  - 高速缓存:典型访问时间 2(ns)，典型容量<4MB
    - 最常用的高速缓存行放置在CPU内部或者非常接近CPU的高速缓存中
    - 高速缓存命中：当某个程序需要读一个存储字时，高速缓存硬件检查所需要的高速缓存行是否在高速缓存中。如果是，称为高速缓存命中，缓存满足了请求，就不需要通过总线吧访问请求发往内存。高速缓存命中通常需要两个时钟周期。未命中就必须访问内存。有些机器有多级高速缓存
    - 多数由硬件控制，
    - 在多核芯片中，设计师必须确定缓存的位置
  - 主存: 典型访问时间 10(ns)，典型容量1~8GB
    - 存储系统的主力，通常称为随机访问存储器(RAM)，过去有时称之为磁芯存储器
    - 非易失性随机访问存储器：与RAM不同，在电源切断后，该存储器不丢失其内容。
      - 只读存储器(ROM)：在工厂中被编码完毕，再不能被修改
      - 电可擦除可编程ROM(EEPROM)和闪存: 与ROM相反，可擦除和重写。闪存常作为便捷式电子设备中的存储媒介。与磁盘存储器不同的是，如果闪存擦除的次数过多，就被磨损了
    - CMOS，易失性的
  - 硬盘: 典型访问时间 10(ms)，典型容量1~4T
    - 由于磁盘是一种机械装置，所以随机访问数据的时间大约慢了三个数量级。其成本也比RAM低了两个数量级，而容量同时也大了两个数量级
    - 硬盘的存储容量=磁头数x磁道(柱面)数x每道扇区数x每扇区字节数
    - 参考资料：https://zhuanlan.zhihu.com/p/163490198
    - 概念：金属盘片、机械臂、磁道、柱面、扇区(典型值512字节)
    - 固态硬盘并非磁盘，没有可移动的部分，数据存储在存储器(闪存)中。但与磁盘相似断电数据不会丢失
    - 虚拟内存：期望运行大于物理内存。将程序放在磁盘，将主存作为一种缓存，用来保存最频繁使用的部分程序。该机制需要快速地映像内存地址，以便把程序生成的地址转换成有关字节在RAM中的物理地址。映像由CPU中一个称为存储器管理单元(MMU)的部件来完成
    - 磁盘读取和写入数据的时候，磁盘会以扇区为单位进行读取和写入数据，即使电脑只需要某个扇区内的几个字节，也必须一次把这几个字节的数据所在扇区中的全部512字节的数据全部读入内存
    - 新买了的硬盘不能直接使用，必须对它进行分区格式化才能存储数据。格式化分区后，逻辑上每个盘片的每一面都会被分为磁道、扇区、柱面这几个虚拟概念
    - 每个扇区的前后两端都会有一些特定的数据，来构成扇区之间的界线标志，磁头通过这些界线标志来识别中的的扇区
- I/O设备：操作系统除了管理CPU和存储器外还要与I/O设备进行交互，I/O设备分为两个部分
  - 设备控制器：控制器是插在电路板上的一块芯片或一组芯片，这块电路板为操作系统提供一个接口，并接收操作系统命令物理地控制设备
  - 设备本身
  - 每个设备控制器都有少量用于通信的寄存器
  - 每类设备控制器都是不同的，所以需要不同的软件进行控制。专门与控制器对话，发出命令并接收响应的软件称为设备驱动程序
  - 每个控制器厂商必须为所支持的操作系统提供响应的设备驱动程序。如一台扫描仪会配有OS X、Windows7、Linux设备驱动程序
  - 为了能够使用驱动程序，必须把设备驱动程序装入操作系统中，使其在内核态运行
    - 设备驱动程序装入操作系统的三个途径
      - 途径一：将内核与设备驱动程序重新链接，然后重启系统，许多UNIX系统以该方式工作
      - 途径二：在一个操作系统文件中设置一个入口，并通知该文件需要一个设备驱动程序，然后重启系统。在系统启动时，OS去寻找所需的设备驱动程序并装载。Windows以该方式工作
      - 操作系统能够在运行时接收新的设备驱动程序并且立即安装好，无须重启系统。正在普及，热插拔设备。如：USB和IEEE 1394设备动态装载
  - 实现输入和输出的三种方式
    - 忙等待：
      - 1. 用户程序发出一个系统调用
      - 2. 内核将其翻译成一个对应设备驱动程序的过程调用
      - 3. 设备驱动程序启动I/O并在一个连续不断的循环中检查该设备，看是否完成了工作(一般有一些二进制位用来指示设备仍在忙碌中)
      - 4. 当I/O结束后，设备驱动程序吧数据送到指定的地方(如果需要)，并返回
      - 5. 操作系统将控制返回给调用者
      - 缺点：占据CPU，CPU一直轮询设备直到对应的I/O操作完成
    - 中断：
      - 设备驱动程序启动设备并且让设备在操作完成时(设备驱动程序检测该设备是否操作完毕)发出一个中断(P33 图1-11)
      - 如果关闭中断时，已有多个设备发出了中断，中断控制器将根据事先赋予每个设备的静态优先级决定先处理那个中断。最高优先级的设备赢得竞争，其他设备则必须等待
    - 直接存储器访问(DMA)芯片：可以控制在内存和某些控制器之间的位流，而无须持续的CPU干预
- 总线
  - 并行总线架构(传统的PCI)，如32位数据通过32条并行的导线发送
  - 串行总线架构(PCIe)，一条被称为数据通路的链路传递集合了所有位的一条消息
  - 即插即用所做的工作是，系统启动地收集有关I/O设备的信息，集中赋予中断级别和I/O地址，然后通知每块卡所使用的数值，其与计算机的启动密切相关
- 启动计算机
  - 每台计算机有一块双亲板/母板
  - 双亲板上有一个称为基本输入输出系统(Basic Input Output System，BIOS)的程序
  - 在BIOS内有底层I/O软件，包括读键盘，写屏幕，进行磁盘I/O以及其他过程。现在这个程序放在一块闪存RAM中，它是非易失性的
  - 计算机启动，BIOS开始运行
    - 首先，检查所安装的RAM数量，键盘和其他基本设备是否安装并正常响应
    - 接着，开始扫描PCIe和PCI总线并找出连在上面的所有设备，即插即用设备也被记录下来。如果现有设备和系统上一次启动时的设备不同，则新的设备将被配置
    - 然后，BIOS通过尝试存储在CMOS存储器中的设备清单决定启动设备。用户可以在系统刚启动之后进入一个BIOS配置程序，对设备清单进行修改
    - 典型地，如果存在CD-ROM(或USB)，则系统试图从中启动；如果失败，系统将从硬盘启动，启动设备(CD-ROM/硬盘)上的第一个扇区(512 bytes)被读入内存并执行。这个扇区中包含一个对保存在启动扇区末尾的分区表检查的程序，以确定那个分区是活动的。然后从该分区读入第二个启动装载模块(引导程序)。来自活动分区的装载模块被读入操作系统，并启动
      - 主引导记录/主引导扇区MBR：开机必须读取的第一扇区
      - 第一扇区包含一：分区表
      - 第一扇区包含二：引导程序，存放操作系统内核存放的真实位置(常见的boot loader有GRUB和LILO)
      - 引导程序会加载内核，内核来管理计算机的硬件资源
    - 然后，操作系统询问BIOS，以获得配置信息。对于每种设备，系统检查对应的设备驱动程序是否存在。如果没有，系统要求用户插入含有设备驱动程序的CD-ROM或从网络下载。一旦有了全部的设备驱动程序，操作系统就将它们调入内核
    - 然后初始化有关表格，创建需要的任何背景进行，并在每个终端上启动登陆程序或GUI

历史

- 第0代 由英国数学家Charles Babbage设计，由于当时技术的限制，未能使机器正常运转。也没有操作系统。
  - 雇佣了一名Ada Lovelace女性编写程序(第一个程序员)
- 第一代 (1945~1955):真空管和穿孔卡片
  - 真空管：上千根电缆接到插件板连接成电路，以便控制机器的基本功能。没有程序设计语言(也没有汇编)，没有操作系统
  - 穿孔卡片：将程序写在卡片上，然后输入计算机，脱离了插件版
- 第二代 (1955~1965): 晶体管和批处理系统
  - 设计、生成、操作、程序和维护人员之间第一次有了明确的分工
  - 晶体管
    - 被称为大型机，所在专用空调的大房间中，由专业人员操作
    - 运行一个job，程序员使用FORTRAN或汇编语言将程序写在纸上，然后穿孔成卡片，再将卡片盒带入输入室，交给操作员，直到输出完成，又打印机打印出结果，操作员送出输入室交给程序员
  - 批处理系统
    - 输入室收集全部作业，用一台相对便宜的计算机将它们读到磁带上(IBM 140计算机适用于读卡片、复制磁带和输出打印，但不适用于数值计算)。另外用较昂贵的计算机(如IBM 7094)来完成真正的计算
  - 主要用于科学与工程计算(如：偏微分方程)。典型操作系统FMS
- 第三代 (1965~1980): 集成电路和多道程序设计
  - 为了满足科学计算与商用，IBM设计了System/360，第一个采用(小规模)集成电路(IC)的主流机型
  - 使用了多道程序设计，在I/O等待时，可以充分利用CPU(解决了要同时满足CPU操作密集的科学计算和I/O操作密集的商业数据处理中CPU空闲浪费的问题)
    - 解决方案：内存复用，将内存分成几部分，每一部分存放不用的作业。当第一个作业等待I/O操作完成时，另一个作业可以使用CPU。(需要特殊硬件来保护作业，避免作业的信息被窃取或收到攻击)
  - 支持同时的外部设备联机操作(SPOOLing)
    - 卡片被拿到机房后能够很快地将作业从卡片读入磁盘。任何时刻当一个作业运行结束时，操作系统就能将一个新作业从磁盘读出，装进空出来的内存区域运行
  - 促使小型机的崛起，并产生了UNIX操作系统
    - UNIX主要有System V和BSD两个版本，IEEE为了使编写的程序能够在任何版本的UNIX上运行，提出了一个UNIX标准，称为POSIX
- 第四代 (1980年至今): 个人计算机
  - 随着LSI(大规模集成)电路的发展，在每平方米的硅片芯片上可以集成数千个晶体管，个人计算时代到来了
- 第五代 (1990年至今): 移动计算机
  - 第一台真正的移动电话出现在1946年，但是重达80斤
  - 第一台真正的手持电话出现在20世纪70年代，大约两斤重
  - 第一台真正的智能手机知道20世纪90年代才出现，诺基亚N9000(手机和个人数字助理)将通话和计算合二为一
  - 1997年，爱立信为其GS88 “Penelope”手机创造出术语智能手机

计算机硬件介绍

系统调用

- 为什么存在系统调用
  - 操作系统具有两种功能:为用户程序提供抽象和管理计算机资源
- 系统调用的过程
  - 任何单CPU计算机一次只能执行一条指令
  - 如果一个进程正在用户态运行一个用户程序，那么它就必须执行一个陷阱或系统调用指令，将控制转移到操作系统
  - 操作系统接着通过参数检查找出所需要的调用进程，然后执行系统调用
  - 把控制返回给在系统调用后面跟随着的指令
  - 系统调用进入内核，而过程调用则不能
- 以read C系统调用举例
  - 调用程序首先把参数压进堆栈
  - 实际调用可能使用汇编语言编写的库过程，一般把系统调用的编号放在操作系统所期望的地方，如寄存器
  - 执行一个TRAP指令，将用户态切换到内核态，并在内核中一个固定地址开始执行
  - 跟随在TRAP指令后的内核代码开始检查系统调用编号，然后分派给正确的系统调用处理器
  - 一旦系统调用处理器完成其工作，控制可能会跟随TRAP指令后面的指令中返回给用户空间库过程
  - 用户程序必须清除堆栈
- 常用的部分POSIX系统调用的库过程
  - 进程管理
    - fork:创建子父进程想用的子进程
    - waitpid:等待一个子进程终止
    - execve:替换一个进程的核心映像
    - exit:终止进程执行并返回状态
  - 文件管理
    - open:打开一个文件供读、写或两者
    - close:关闭一个打开的文件
    - 把数据从一个文件读到缓冲区中
    - write:把数据从缓冲区写到一个文件中
    - lseek:移动文件指针
    - stat:取得文件的状态信息
  - 目录和文件系统管理
    - mkdir:创建一个新目录
    - rmdir:删去一个空目录
    - link:创建一个新目录项
    - unlink:删去一个目录项
    - mount:安装一个文件系统
    - umount:卸载一个文件系统
  - 杂项
    - chdir:改变工作目录
    - chmod:修改一个文件的保护位
    - kill:发送信号给一个进程
    - time:自1970年1月1日起的流逝时间

基本概念

- 进程
  - 概念
    - 进程本质上是正在执行的一个程序
  - 相关内容
    - 地址空间(最小到最大存储位置的列表)
      - 存放可执行程序
      - 存放程序的数据
      - 存放程序的堆栈
    - 资源集
      - 寄存器(程序计数器和堆栈指针)
      - 打开文件的清单
      - 突出的报警
      - 有关进程清单
      - 运行程序所需要的其他信息
    - 进程表(数组或链表结构)
      - 进程表是位于操作系统中的一张表，用于存放进行的某些信息
      - 当前存在的每个进程都要占用其中一项
      - 进程表中存放除了该进程自身地址空间的内容以外的与进程有关的所有信息
  - 总结
    - 进程基本上是容纳运行一个程序所需要所有信息的容器
    - 被挂起的线程(操作系统分配的时间片用完了)包括两个部分
      - 进程的地址空间
      - 对应的进程表项(包括寄存器以及稍后重启动该进程所需要的许多其他信息)
  - 进程管理
    - 创建进程的系统调用
    - 终止进程的系统调用
    - 若一个进程能够创建一个或多个进程(子进程)，而这些进程又可以创建子进程，则很容易得到进程树
    - 进程间通信，进程间彼此通信以便同步它们的行为
    - 其他可用的进程系统调用：申请/释放需要/不需要的内存
    - 系统管理器授权每个进程使用一个给定的UID，UNIX中，有一个UID称为超级用户
- 地址空间
  - 从某个最小值的存储位置(通常是零)到某个最大值的存储位置的列表，在这个地址空间中，进程可以进行读写
  - 进程有比主存大的地址空间
    - 许多32位或64位地址的计算机中，分别有2的32次方或2的64次方字节的地址空间
    - 早期的计算机只能“认命”，进程加载不起来
    - 现在可以使用虚拟内存技术，操作系统可以把部分地址空间装入主存，部分留在磁盘上，并且在需要时来回交换它们
  - 本质
    - 操作系统创建了一个地址空间抽象，作为进程可以引用地址的集合。该地址空间与机器的物理内存解耦
    - 对地址空间和物理空间的管理是操作系统功能的一个重要部分
- 文件
  - 操作系统的一项主要功能是隐藏磁盘和其他I/O设备的细节特性，给程序员提供一个良好、清晰的独立于设备的抽象文件模型
  - 系统调用：创建文件、删除文件、读文件和写文件
  - 文件系统：提供文件保存的地方，大多数操作系统支持目录的概念，从而可把文件分类成组
    - 文件树
      - 树状结构可以有很多层，四层、五层、更多层
      - 目录层次可能存在数年之久
    - 文件树与进程树的区别
      - 进程树结构层次不深，很少超过三层
      - 进程树是暂时的，最多几分钟
      - 父进程能控制和访问子进程
      - 所有权及保护方面也有区别
      - 文件所有者之外的其他用户也可以访问文件
  - 每个进程有一个工作目录，对于没有以斜线开头给出绝对地址的路径，将在这个工作目录下寻找
  - 在读写文件前，首先要打开文件，检查其访问权限。若权限许可，系统将返回一个小整数，称作文件描述符，供后续操作使用。若禁止访问，系统则返回一个错误码
  - UNIX中的两个重要概念
    - 安装文件系统：为了提供一个出色的方式处理可移动介质，UNIX允许把光盘上的文件系统接到主文件树上。mount之前两个是分离且无关的，mount系统调用后，把CD-ROM上的文件系统接到程序所希望的主文件树上(P40 看图1-15)
    - 特殊文件
      - 块特殊文件
      - 字符特殊文件
  - 管道：管道是一种虚文件它可连接两个进程，从而实现进程间的通信
- 输入/输出
  - 输入和输出设备包括键盘、显示器、打印机等。操作系统对这些设备进行管理
- 保护
  - 文件保护
    - UNIX操作系统通过对每个文件赋予一个9位的二进制保护代码，对UNIX中的文件实现保护
    - 一个用于所有者，一个用于与所有者同组，一个用于其他人
    - 该保护代码有三个3位字段：每个字段中有一位用于读访问，一位用于写访问，一位用于执行访问(rwx位)
  - 保护系统不被人类或非人类(病毒)入侵
- shell
  - shell是终端用户与操作系统之间的接口
  - 用户登录时，同时启动了一个shell，以终端作为标准输入和标准输出
  - 用户键入一个命令，如date
  - shell创建一个子进程，并运行date程序作为子进程。在该子进程运行期间，shell等待它结束。子进程结束后，shell再次显示提示符，等待下一个输入
  - 如果用户在命令后加上一个&符号，则shell将不等待器结束，而直接显示出提示符

UNIX中进程将其存储空间划分为三段

- 堆栈段
  - 堆栈在使用时自动向中间增长
  - 堆栈段数据向上增长而堆栈向下增长，夹在中间的是未使用的地址空间堆栈段
- 数据段(如变量)
  - 数据段的扩展是显示地通过系统调用brk进行
- 正文段(如程序代码)

XMind 试用模式