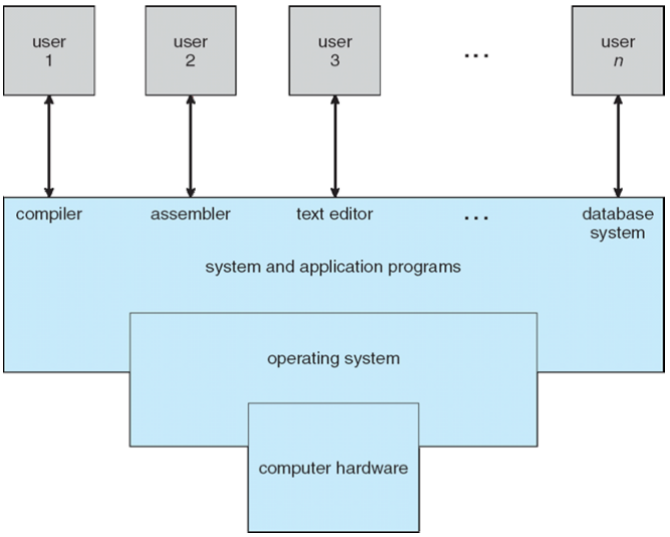
OS概述

笔记源文件：Markdown，长图，PDF，HTML

1. 操作系统概念

1.1. OS概述：为何物

0️⃣计算机系统部件：硬件(CPU/内存/输入输出设备/系统总线)、操作系统、应用程序以及用户



1️⃣为何需要操作系统：方便用户交互与使用，高效管理硬件

2️⃣**定义：控制和管理计算机系统资源，方便用户使用的程序和数据结构的集合**

1.2. 操作系统的特点

1.2.1. 并发

1️⃣含义：计算机中同时存在多个运行的程序，进程由此引入

2️⃣实现方式：程序分时交替执行，一段时间内并发执行，一个时刻只有一个程序在执行

3️⃣与并行的辨析：并行是同一时刻(而非同一时间但)执行多个程序

1.2.2. 共享

1️⃣含义：系统资源可供内存中多个并发执行进程使用

2️⃣共享方式之一：互斥共享方式(如打印机)

1. 含义：一段时间内只许一个进程访问该资源，别的进程要访问只能先等
2. 临界资源：一段时间只允许一个进程访问的资源

3️⃣共享方式之二：同时访问方式(如磁盘)，指宏观上多个进程同时访问一个资源(微观上分时交替)

1.2.3. 虚拟

1. 含义：物理实体映射为若干个对应的逻辑实体(用户能感觉到的)
2. 虚拟处理器(时分复用技术)：让多道程序并发执行，分时使用一个处理器，让每个用户觉得自己独享CPU
3. 虚拟存储器(空分复用技术)：利用软件技术将辅存扩充为主存
3. 虚拟IO设备：将一台I/O虚拟为多台逻辑I/O，允许每个用户占用一台逻辑I/O

1.2.4. 异步(不确定性)

进程的执行顺序/时间，不是一贯到底而是以不可预测的方式走走停停

1.3. 操作系统功能

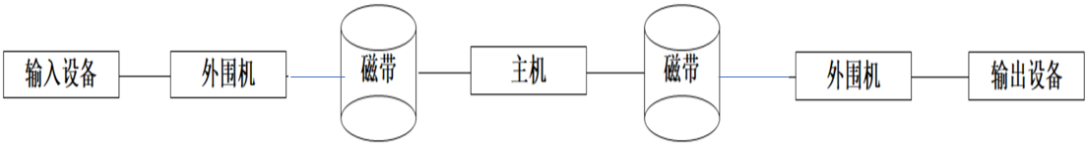
1. 管理系统资源：CPU(进程)管理+内存管理+文件管理+设备管理
2. 作为用户接口，主要分为命令接口(联机+脱机)，程序接口
    1. 联机命令接口(分时系统)：用户通过控制台/终端输入命令→系统解释/执行命令行→控制权转回终端
    2. 脱机命令接口(批处理系统)：用户不直接干预作业运行，事先用控制命令写成一份作业操作说明，连同作业一起提交给系统
    3. 程序接口：供用户在程序中使用是的一堆系统调用，通过GUI(图像用户界面)调用程序接口
3. 扩充计算机资源：没操作系统的计算机叫裸机，覆盖了软件的机器称为扩充机器/虚拟机

2. 操作系统的发展：分时操作系统前

无OS(真空管)→批处理OS(晶体管)→多道程序(集成电路)→分时系统(大规模集成电路)

2.1. 无操作系统阶段

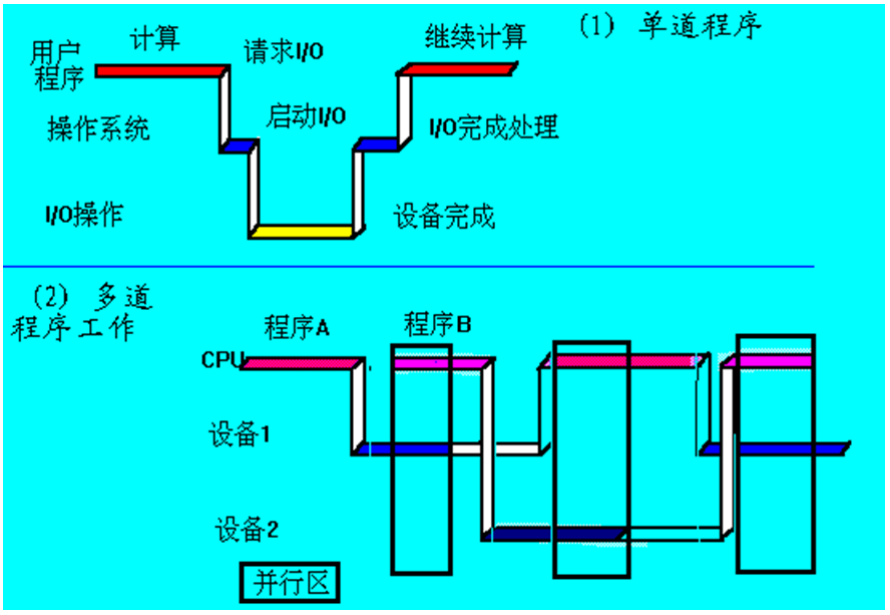
1. 概述：工作方式(手搓)+编程语言(机器语言)+IO(打纸袋)
2. 缺点：用户独占全机利用率低，CPU等待用户(装卸纸带)
3. 脱机IO：将IO操作放在别的计算机上实现，让磁带作为中介



2.2. 单道批处理系统(操作系统前身)

1. 系统工作方式：作业脱机输入磁带，OS配置监督程序，监督程序一个个把磁带中的作业调入内存处理完后调出
2. 特点：自动性(作业自动进入)，顺序性(作业进入内存后顺序执行)，单通道性(内存中只能同时运行一个程序)
3. 一大缺陷：CPU和I/O设备使用忙闲不均

2.3. 多道批处理系统



1. 工作方式：区别于单道，作业排队(一个/多个一起被操作系统)送入内存，内存中可以有多个独立作业，但同时运行/共享资源(CPU，IO等)

2. 特点：多道性(内存中多道程序)+无序性(作业完成顺序与进入内存顺序无关)+调度性(作业/进程调度)

3. 缺点：无交互能力，用户响应时间长，作业平均周转时间长(有可能第一个进入，最后一个出)

3. 正式操作系统类型

pre.作业的概念：操作系统执行一个工作所作的工作(程序，数据，命令)集合

3.1. 分时OS

1. 分时技术：把CPU运行时间分为时间片，按时间片轮流把CPU给各联机作业使用。若某作业在其时间片内不能完成其计算，则下一轮时间片再计算行

2. 实现方法种类：

    1. 简单分时操作系统(内存中只有一个作业，但是时间片满后就被踢出)
    2. 有先后台的分时操作系统(把作业划分为前后台，前台同于简单分时/后台批处理，前台运行后台才能运行)
    3. 多道分时操作系统(引入多道程序技术，内存可同时装入多道作业)

3. 特点：多路性(一设备多终端，每个终端的用户同时使用)，交互性(用户直接控制程序运行)，独占性(用户都以为自己独占计算机)，及时性(计算机快速响应用户需求)

3.2. 实时OS(专用OS)

在极短时间内(短于分时系统)对外界信息做出响应，实时性高，专用性强

1. 实时控制系统(硬件)：以计算机为中心的生产过程控制系统(如数控机床)，实时采集/处理数据

2. 实时信息处理系统(软件)：及时接收远程终端的服务请求，并快速响应(如12306系统)

3.3. 其它OS

1. 嵌入式OS：在各种设备/装置/系统中(比如电器)，完成特定功能的软硬件系统
2. 并行系统(紧耦合)：有紧密通信的(不要理解为多通道)、多于一个CPU的多处理器系统
3. 网络操作系统：通过网络将多个计算机系统互连来交换信息/共享资源/协作处理，特点为：
   1. 系统中的计算机系统在物理上是分散的，自治的(都有各自系统)
   2. 系统互连要通过通信设施(硬/软件)来实现
4. 分布式系统(松散耦合)：分散处理单元经网络连城统一OS，可将大任务划分并分配
   1. 特点：统一(它是个统一OS)+共享性(所有资源共享)+透明性(用户眼里不知道哪个计算机在处理请求)+自治性(多个主机都处于平等地位)
   2. 优点：成本低性能高，可靠(一个CPU挂了照样运行)

3.4. 番外：分布式OS vs 网络OS

1. 分布式系统是OS同质(OS一样)，网络OS要求协议同质(OS可不一样但是网络协议要一样)
2. 分布式OS可将进程分散在多个主机，网络OS进程则不能迁移
3. 分布式OS中用户不知道哪个主机在处理任务，网络OS则反之
4. 分布式系统容错率更大

4. OS运行环境

4.0. 硬件保护

1. IO保护：I/O都是特权指令(用户不能直接I/O)，必须通过系统调用，需要保护IO必须先保护中断向量(在内存中)，所以必须保护内存
2. 内存保护：确定进程能访问空间，使用基址+界限寄存器(内存开始+长度)，这两个寄存器只能由特权指令在核心态加载
3. CPU保护：防止用户程序死循环/不调用系统服务/不将控制权返回OS，设置定时器

PS: 定时器，一定时间后就会在中断，把控制权返回OS

4.1. CPU运行模式

4.1.1. 两种模式

1. 核心态(管态/系统态)：OS管理程序执行时机器处于核心态，特权高，可执行一切指令，可访问所有寄存器/存储区
2. 用户态(目态)：用户程序执行时机器处于用户态，特权低，只执行规定命令，只访问特定寄存器/存储区

PS: **一开机时处于管态，执行用户程序时转为用户态；出现中断/陷阱时，硬件会切换回管态**

3. 两种状态的程序被严格分开存储，在CPU中以不同方式执行
4. 用户态程序如何调用核心态程序：去执行访问核心态命令→中断→中断系统转入OS内相应程序

4.1.2. 有关概念

1️⃣特权指令：只给OS内核使用，只有在核心态中才能使用，如IO/中断屏蔽/存储保护/清内存/设置时钟

2️⃣OS内核：硬件强关联模块(时钟管理/中断处理/驱动)+运行频率高的程序(进程/存储/设备管理)

1. 时钟管理(最关键)：向用户提供标准系统时间，管理时钟中断来切换进程
2. 中断机制：小部分由内核负责(保护/恢复中断现场信息，转移控制权到相关程序)
3. 原语：关闭中断的公用小程序，是最接近硬件的软件，运行时间段调度频繁，一条原语必须一次性执行完
4. 系统控制的数据结构及处理：登记状态信息的数据块(作业控制块/进程控制块等)+如何操作这些块(进程管理等)

4.2. 中断和异常的处理

1️⃣硬件中断：设备控制器利用中断通知CPU它已经完成了某个操作

2️⃣软件中断：也称为陷阱，包括异常(Exception)与系统调用(System call)

PS：中断会将控制权转移到中断服务程序，通用程序检查是否中断，不同终端有不同代码处理

3️⃣中断向量表：每一类中断对应一个中断向量，合在一起就是中断向量表

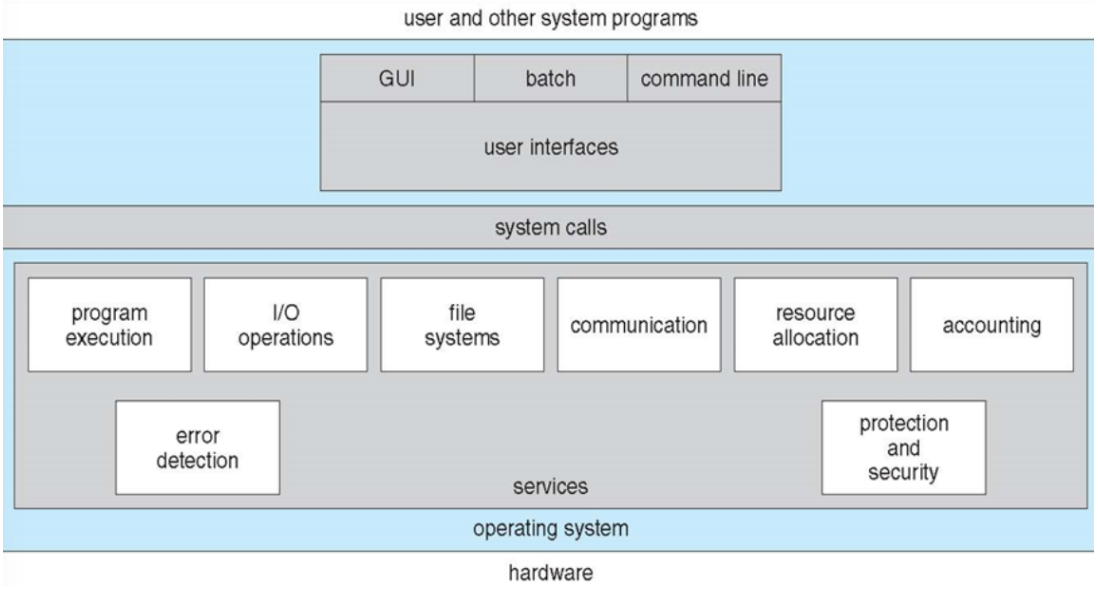
中断向量	中断号	中断用途
00 ~ 03	0	除法溢出中断
04 ~ 07	1	单步(用于DEBUG)
08 ~ 0B	2	非屏蔽中断(NMI)
0C ~ 0F	3	断点中断(用于DEBUG)
10 ~ 13	4	溢出中断
14 ~ 17	5	打印屏幕
18 ~ 1F	6,7	保留

4️⃣IO中断：

1. IO操作：分为同步(OS干等IO结束)+异步(OS在IO时去干别的)
2. IO操作完后就触发一个中断

5. OS结构

5.1. 操作系统服务一览



5.2. 系统调用

六大类：进程控制/文件管理/设备管理/信息维护/通信/保护

5.2.1. 系统调用概念

1. 功能：提供在运行程序和操作系统之间的接口，属于软中断(aka陷入/异常指令/访管指令)
2. 途径：程序通过API(高级应用程序接口)访问
3. 常见API：Win32 API，POSIX API(UNIX/Linux/MacOS)，用于Java虚拟机(JVM)的Java API
4. 系统调用处理机构：陷入/异常处理机构(TRAP)

5.2.2. 系统调用的过程

1. 大致流程：系统调用把应用程序请求传给内核→内核调用内核函数完成处理→执行结果返回应用程序

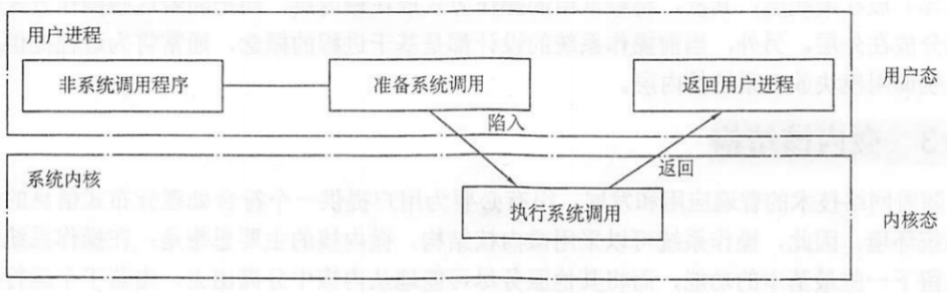
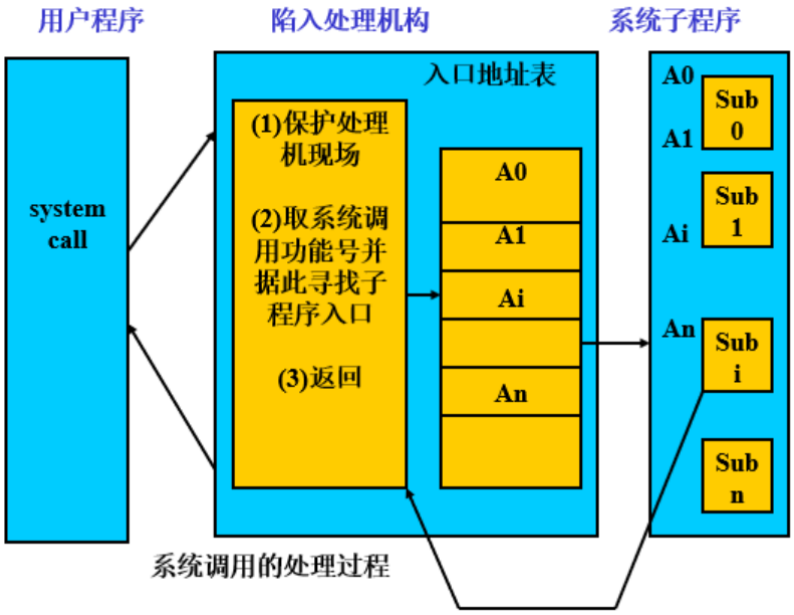


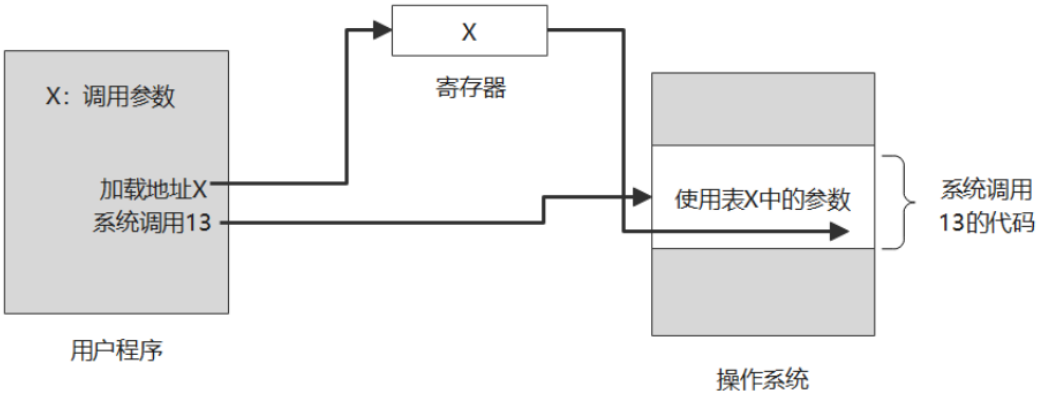
图 1-6　操作系统执行系统调用的流程

2. 实现细节：用户调用→CPU中断/发信给陷入处理机构→陷入处理机构保护现场/通过入口地址表找到系统调用程序→执行完系统程序→CPU恢复现场



5.2.3. 向OS传参的方式

用寄存器+用栈(程序压入/OS取出)+通过参数表(下图)



参数在表中，表的地址通过寄存器传递

5.3. 操作系统体系结构类型

5.3.1. 模块组合结构(UNIX，MS-DOS)

1 含义：操作系统是一个整体模块，由若干模块按照一定结构组成

2 利弊：利在于结构紧凑/接口简单/效率高，弊在于模间调度混乱/接口设计灵活度低/不适用于大型系统

5.3.2. 层次结构

1 含义：将所有模块按功能调用次序排列成若干层，模块间只存在单向调用/依赖

2 优点：

1. 低层和高层可分别实现(可扩充)
2. 高层错误不会影响到低层，便于调试、利于功能的增删改
3. 调用关系清晰(高层对低层单向依赖)，避免递归调用，有利于保证设计和实现的正确性

3 缺点：难以解决OS各模块应放哪一层(当下将为进程有关调用模块放在内层)

5.3.3. 微内核结构：用户程序(客户/服务器进程)+微内核

1️⃣含义：OS Kenel中只留基本功能(任务管理/虚存管理/进程间通信)，其他功能分出去，由服务器进程完成其他功能，形成CS模式

2️⃣工作方式：C进程通过内核请求S进程，然后C进程获得操作系统服务

3️⃣优点：可靠(某个服务器进程崩了不会全崩)，灵活(接口灵活增减)，便于维护(修改服务器代码不会崩)，适用于分布式系统

4️⃣缺点：效率不高，通信频繁

5.3.4. 模块化内核(可加载内核模块)

1️⃣面向对象：每个内核模块都封装抽象，保留功能接口

2️⃣核心组件分离：核心组件(文件系统/网络协议栈/设备驱动)成为独立模块，模块只在需要时加载

3️⃣模块间通信：通过定义接口

4️⃣动态加载：模块不是在内核启动时加载，而是在内核运行时按需加载，比如某个硬件挂在后相应驱动模块才加载

5.4. 虚拟机

软件实现，模拟硬件系统的功能，安装并运行操作系统和应用程序