Exam OS Learning Note

考研 OS 学习

ElegantLaTeX Program

Victory won't come to us unless we go to it.

作者：Miao

时间：August 19, 2023

邮箱: chenmiao.ku@gmail.com

版本: 0.10

目　　录

第 1 章 计算机系统概述

1.1 操作系统的基本概念

1.1.1 操作系统的概念

操作系统 *(Operator System)* 是指控制和管理整个计算机系统的硬件和软件资源，合理地组织、调度计算机的工作与资源的分配，进而为用户和其他软件提供方便接口与环境的程序集合。

操作系统是计算机系统中最基本的系统软件。

1.1.2 操作系统的特征

操作系统有如下四大特征：

1. 并发 (Concurrence)

   并发是指两个或多个事件在同一时间间隔内发生。值得注意的是：并发和并行是两个不同的概念，并行一定并发，并发不一定并行。对于单处理器来说，只能并发执行。

   并发是 *OS* 最为基础且必要的特征。

2. 共享 (Sharing)

   共享即资源共享，是建立在并发之上的。

   (互斥式共享) 系统中的某些资源在一个时间段内有且仅能一个程序进行访问和操作，且把该资源称为临界资源。

   (同时式共享) 该类资源可以由多个程序同时段进行访问 (多出现在读操作上)

   并发和共享是操作系统最基本的特征，两者之间互为存在的条件：资源共享必定是由并发产生的，而共享影响了并发则会导致并发崩溃。

3. 虚拟 (Virtual)

   虚拟是指把一个物理上的实体变为若干逻辑上的对应物。物理实体是实际存在的，而虚拟逻辑是用户感觉上的事物。

   虚拟处理器的存在：时分复用技术

   虚拟内存的存在：空分复用技术

4. 异步 (Asynchronism)

   多道程序环境允许多个程序执行，但由于资源有限 (竞争的情况)，进程并非一贯到底的执行。而且走走停停，不停的切换，因此以不可预知的速度向前推

进。

1.1.3 操作系统的目标与功能

为了给多道程序提供环境，*OS* 应该具备：处理机 *(*进程*)* 管理、存储器 *(*内存*)* 管理、设备管理和文件管理 *(*现代操作系统中，不仅仅这几个模块*)*。

除了上述的模块外，还需要提供各种接口。

操作系统作为计算机系统资源的管理者

1) 处理机 (进程) 管理

在多道程序环境下，处理机的分配和运行都以进程 (或线程) 为基本单位，因此对处理机管理可归结为进程管理。

并发指的是计算机内同时运行多个进程，因此如何管理进程则是最主要的任务。进程管理的主要功能包括进程控制、进程同步、进程通信、死锁处理、处理机调度等。

2) 存储器 (内存) 管理

内存管理是为了给多道程序的运行提供良好的环境，方便用户使用以及提高利用率。其主要包括内存分配与回收、地址映射、内存保护与共享和内存扩充等。

3) 文件管理

计算机中的信息都是以文件的形式存在的，文件管理包括文件存储空间的管理、目录管理以及文件读写管理与保护等。

4) 设备管理

设备管理的主要任务是完成用户的 I/O 请求，方便用户使用各种设备，提高设备利用率。主要包括缓冲管理、设备分配、设备处理和虚拟设备等。

操作系统作为用户与计算机硬件系统之间的接口

1) 命令接口

(联机命令接口 (交互式)) 适用于分时或实时操作系统。由一组键盘命令组成，用户通过控制台或 *terminal* 输入命令与 *OS* 进行交互。类似于 Python 交互器。

(脱机命令接口 (批处理)) 适用于批处理系统。由一组作业控制命令组成，不能直接干预作业的运行，事先使用对应的作业控制命令写成一份操作流程，然后递交给 *OS*。类似于 Bash 脚本。

2) 程序接口

程序接口由系统调用 (又称广义指令) 组成，例如 printf、malloc 等 C 语言调用接口。

GUI 就是图形接口，其最终是通过调用程序接口实现的。严格的说，GUI 不是操作系统的一部分，但 GUI 所使用的系统调用是 OS 的一部分。

操作系统实现了对计算机资源的扩展

没有任何软件支持的计算机被称为裸机，其仅构成计算机系统的物质基础。

也就是说，操作系统的内部是各种物理结构所组成的逻辑环境，操作系统所提供的资源管理功能和方便用户的各种服务功能，将裸机改造成功能更强、更方便的机器。通常，把覆盖了软件的机器称为扩充机器或虚拟机。

1.2 操作系统发展历程

1.2.1 手工操作系统

此阶段并未产生严格意义上的 *OS*。

手工操作系统的突出缺点：用户独占全机，资源利用效率极低；*CPU* 等待手工操作，*CPU* 利用效率极低。

1.2.2 批处理阶段

为了解决人机矛盾以及 *I/O* 设备之间速度不匹配的问题，出现了批处理系统。

单道批处理系统

系统对作业的处理是成批进行的，但内存中始终只保存一道作业。其主要特征为：

1) 自动性。磁带上的一批作业自动逐个进行，无需人工干预
2) 顺序性。磁带上的作业按顺序进入内存
3) 单道性。内存中仅有一道作业运行，即监督程序每次从磁带上只调入一道程序。

单道批处理系统的主要问题在于：内存中仅有一道作业，*CPU* 有大量的时间是在等待 *I/O* 的完成。

多道批处理系统

为了解决资源利用率和系统吞吐量，引入了多道程序技术。多道程序技术允许多个程序同时进入内存并允许在 *CPU* 中交替运行，共享系统中的各种软硬件资源。

其设计的特点为多道、宏观上并行，微观上串行。但是，多道程序设计需要解决：

1) 如何分配处理器
2) 如何分配内存
3) 如何分配 I/O 设备

4) 如何组织和存放大量的程序和数据，且保证数据安全和一致性

其优点在于：资源利用率高，且共享计算机资源从而使各种资源得到充分利用；系统吞吐量大，*CPU* 和其他资源保持“忙碌”状态。

缺点在于：用户响应时间较长；不提供人机交互能力，用户既不能了解运行情况也不能控制计算机。

分时操作系统

分时技术，也就是将处理器的运行时间分成很短的时间片，按照时间片轮流把处理器分配给各联机作业使用。这使得每个用户 *(*程序*)* 感觉起来就像自己独占一台计算机。

分时操作系统，指的是多个用户通过终端同时共享一台主机，用户可以同时与主机进行交互操作而互不干扰。因此，分时系统最关键的问题在于：如何使用户与自己的作业交互。分时系统支持多道程序设计，但不同于多道批处理，分时系统是人机交互的系统：

1) 同时性。同时性又称多路性，允许多个终端用户同时使用一台计算机。
2) 交互性。用户能够与系统进行人机对话。
3) 独立性。系统中的各个用户独立操作，互不干扰。
4) 及时性。用户请求能在很短时间内响应，采用时间片轮转算法。

实时操作系统

为了满足某个时间限制内完成某些紧急任务而不需要时间片排队，诞生了实时操作系统。

硬实时系统：某个动作必须绝对地在规定的时刻内完成或发生。

软实时系统：能够接收偶尔违反时间规定且不会引起任何永久性的损害。

在实时操作系统的控制下，计算机系统接收到外部信号后及时处理，并在严格时间内处理完毕，其主要特点就是及时性和可靠性。

1.3 操作系统运行环境

1.3.1 处理器运行环境

在 OS 中，CPU 通常执行两种不同性质[1]的程序：一种是 OS Kernel 程序；一种是 User program。对于 OS 来说，前者是后者的管理者，管理程序需要执行一些特权指令，而用户程序出于安全考虑不允执行。

[1]就我所知，现代 *OS* 中的等级应该大多数是三级，*M*、*S*、*U*，分别表示机器级、监管者、用户级。

同时，OS 中将程序的环境分成了两态，分别对应了特权级的内核态以及非特权级的用户态[1]，其用于用户程序和内核程序不同运行的环境。

1) 特权指令，指的是不允许用户直接使用的指令，比如 *I/O* 指令，置中断指令等一系列需要内核态环境下完成的指令。
2) 非特权指令，指的是允许用户直接使用的指令，不能直接访问系统中的软硬件资源，仅限于访问用户的地址空间。

需要注意的是：*OS 中的内核态和用户态是不断切换的，以适应不同的需求*。如果想要从内核态切换到用户态，那么直接使用对应的特权指令进行“降级操作”(例如在 rv 架构上的 sret 指令)。如果需要从用户态切换到内核态，那么必须通过中断操作 (此处的中断是广义的，包括了狭义的中断和异常)，触发中断信号，硬件自动完成变态操作。

内核是计算机上配置的底层软件，管理了系统的各种资源：

1. 时钟管理

   在计算机的各部件中，时钟是最为关键的设备。时钟的第一功能是计时，OS 通过时钟管理来获取标准的系统时间，且通过时钟中断实现进程的切换。

2. 中断机制

   引入中断的初衷是为了提高多道程序运行环境中 CPU 的利用率。就目前来说，现代操作系统就是靠中断驱动的软件。

   中断机制中，仅有一小部分属于内核，负责保护和恢复终端现场信息，转移控制权到相关的处理程序。

3. 原语

   操作系统提供的基本操作接口或函数，用于实现高级操作系统功能和管理底层资源。原语通常是操作系统内核的一部分，提供了对硬件和系统资源的底层访问和控制。

   1) 处于 OS 的最底层，是最接近硬件的部分
   2) 具有原子性，其操作只能一气呵成
   3) 运行时间短，调用频繁

   定义原语的直接方法是关闭中断，让其所有动作不可分割地完成后再打开中断。

4. 数据结构及处理

   在 OS 中，定义了很多登记状态的数据结构，如作业控制块、进程控制块 (PCB)、设备控制块、各种链表、队列等。常见的为三种：

   1) 进程管理。进程状态、调度与分派、创建、撤销等
   2) 内存管理。内存分配回收、保护、对换等
   3) 设备管理。缓冲区、设备分配、回收等

[1]在 OS 中，一般会存在一个特殊寄存器 PSW(程序状态字寄存器)，主要用于判断当前运行环境是内核态还是用户态。

1.3.2 中断和异常的概念

OS 引入内核态和用户态后，必须考虑的事情就是如何切换，如何留下“门”。

中断和异常的定义

中断 (Interruption) 也叫做外中断，指的是来自 *CPU* 执行指令外部的事件，通常用于信息输入/输出，例如 *I/O* 中断，时钟中断等。

异常 (Exception) 也叫做内中断，指的是 *CPU* 执行指令内部的事件，例如非法操作、越界、缺页或 *trap* 指令。值得注意的是，异常不能被屏蔽，一旦出现，需要立刻处理。

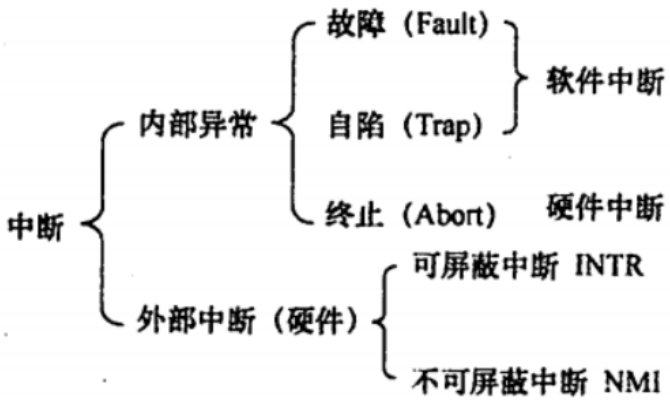

图 1.1: 中断和异常的联系与区别

中断和异常的分类

外中断可分为可屏蔽中断和不可屏蔽中断。可屏蔽中断通过 *INTR* 线发出中断请求，通过改变屏蔽字可以实现多重中断；不可屏蔽中断通过 *NMI* 线发出中断请求，一般比较紧急，因此不可屏蔽。

异常分为故障、自陷 (trap) 和终止。故障 (Fault) 通常由指令引起的异常，如非法操作码，缺页异常，除零等；自陷 (Trap) 是一种实现安排的“异常”行为，用户通过自陷操作从而进入内核态执行特权代码。终止 (Abort) 指出现了使 CPU 无法继续执行的硬件故障。

故障异常和自陷异常属于软中断，外部中断和终止异常属于硬中断。

中断和异常的处理

1. 当 CPU 运行时检测到异常事件 (或检测到一个中断请求信号)
2. 保存现场状态

3. CPU 打断当前执行，通过中断 (异常) 信号，查询中断向量表，跳转到对应的中断或异常处理程序
4. 若能处理，则返回执行，恢复现场; 若不能处理，终止程序

1.3.3 系统调用

用户在程序中调用 *OS* 所提供的一些子功能，系统调用可被视为特殊的公共子程序。

系统中的各种共享资源都是由 *OS* 统一管理，因此，*OS* 为了安全以及稳定起见，统一的为用户提供了接口，使得用户只要涉及资源相关的操作，就必须通过系统调用来提出服务请求，由 *OS* 来代理完成。

- 设备管理。完成设备的请求和释放，以及设备的启动等
- 文件管理。完成文件的读写、创建、删除等
- 进程管理。完成进程的创建、删除、阻塞等
- 进程通信。完成进程之间信息的传递等
- 内存管理。完成内存的分配、回收等

那么，对于上述的操作，其内部肯定是需要在内核态中运行一些特权指令的，因此，系统调用就需要用用户态切换到内核态去执行。因此，OS 提供了特殊的 trap 指令来发起系统调用请求。

也就是说，OS 的运行环境就可以为：用户通过 OS 运行上层应用，其依赖于 OS 的底层管理程序提供服务支持，当需要管理程序服务时，则通过中断 (异常) 机制进入内核态，运行管理程序。同时，在进入内核态时，需要保存现场。

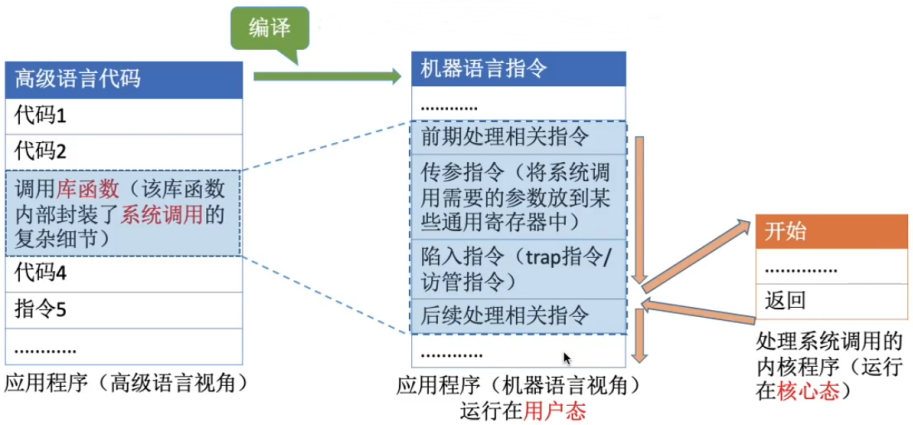

图 1.2: 系统调用的过程

值得注意的是：

1. 陷入指令是在用户态执行的，会立即产生一个软中断，使 *CPU* 进入内核态。

2. 发出的系统调用请求在用户态，执行在内核态 *(*通过传参指令，告诉系统调用入口程序要调用哪一个系统调用*)*。