计算机系统

# 一、概述

## 一）系统基本特征

1. 并发：并发是指宏观上在一段时间内能同时运行多个程序，而并行则指同一时刻能运行多个指令。并行需要硬件支持，如多流水线、多核处理器或者分布式计算系统。操作系统通过引入进程和线程，使得程序能够并发运行。

2、共享：共享是指系统中的资源可以被多个并发进程共同使用。有两种共享方式：互斥共享和同时共享。互斥共享的资源称为临界资源，例如打印机等，在同一时间只允许一个进程访问，需要用同步机制来实现对临界资源的访问。

3、虚拟：虚拟技术把一个物理实体转换为多个逻辑实体。主要有两种虚拟技术：时分复用技术和空分复用技术。多个进程能在同一个处理器上并发执行使用了时分复用技术，让每个进程轮流占有处理器，每次只执行一小个时间片并快速切换。虚拟内存使用了空分复用技术，它将物理内存抽象为地址空间，每个进程都有各自的地址空间。地址空间的页被映射到物理内存，地址空间的页并不需要全部在物理内存中，当使用到一个没有在物理内存的页时，执行页面置换算法，将该页置换到内存中。

4、异步：异步指进程不是一次性执行完毕，而是走走停停，以不可知的速度向前推进。

## 二）系统基本功能

1. 进程管理：进程控制、进程同步、进程通信、死锁处理、处理机调度等。

2. 内存管理：内存分配、地址映射、内存保护与共享、虚拟内存等。

3. 文件管理：文件存储空间的管理、目录管理、文件读写管理和保护等。

4. 设备管理：完成用户的 I/O 请求，方便用户使用各种设备，并提高设备的利用率。主要包括缓冲管理、设备分配、设备处理、虛拟设备等。

## 三）系统调用

如果一个进程在用户态需要使用内核态的功能，就进行系统调用从而陷入内核，由操作系统代为完成。

Linux的主要系统调用

| Task | Commands |
| --- | --- |
| 进程控制 | fork(); exit(); wait(); |
| 进程通信 | pipe(); shmget(); mmap(); |
| 文件操作 | open(); read(); write(); |
| 设备操作 | ioctl(); read(); write(); |
| 信息维护 | getpid(); alarm(); sleep(); |
| 安全 | chmod(); umask(); chown(); |

## 四）大内核与微内核

大内核：将操作系统功能作为一个紧密结合的整体放到内核。

微内核：由于操作系统不断复杂，因此将一部分操作系统功能移出内核，从而降低内核的复杂性。移出的部分根据分层的原则划分成若干服务，相互独立。

## 五）中断分类

1. 外中断：由 CPU 执行指令以外的事件引起，如 I/O 完成中断，表示设备输入/输出处理已经完成，处理器能够发送下一个输入/输出请求。此外还有时钟中断、控制台中断等。

2. 异常：由 CPU 执行指令的内部事件引起，如非法操作码、地址越界、算术溢出等。

3. 陷入：在用户程序中使用系统调用。

# 二、进程管理

## 一）进程与线程

进程是资源分配的基本单位。进程控制块 (Process Control Block, PCB) 描述进程的基本信息和运行状态，所谓的创建进程和撤销进程，都是指对 PCB 的操作。

线程是独立调度的基本单位。一个进程中可以有多个线程，它们共享进程资源。

**区别：**

拥有资源：进程是资源分配的基本单位，但是线程不拥有资源，线程可以访问隶属进程的资源。

调度：线程是独立调度的基本单位，在同一进程中，线程的切换不会引起进程切换，从一个进程中的线程切换到另一个进程中的线程时，会引起进程切换。

系统开销：

通信方面：

## 二）进程状态的切换

就绪状态（ready）：等待被调度

运行状态（running）

阻塞状态（waiting）：等待资源

## 三）进程调度算法

不同环境的调度算法目标不同，因此需要针对不同环境来讨论调度算法。

1、批处理系统：先来先服务；短作业优先；最短剩余时间有限；

2、交互式系统：时间片轮转；优先级调度；多级反馈队列；

3、实时系统：实时系统要求一个请求在一个确定时间内得到响应。分为硬实时和软实时，前者必须满足绝对的截止时间，后者可以容忍一定的超时。

## 四）进程同步

临界区：对临界资源进行访问的那段代码称为临界区。为了互斥访问临界资源，每个进程在进入临界区之前，需要先进行检查。

同步与互斥：同步：多个进程按一定顺序执行；互斥：多个进程在同一时刻只有一个进程能进入临界区。

信号量：信号量（Semaphore）是一个整型变量，可以对其执行 down 和 up 操作，也就是常见的 P 和 V 操作。down : 如果信号量大于 0 ，执行 -1 操作；如果信号量等于 0，进程睡眠，等待信号量大于 0；up ：对信号量执行 +1 操作，唤醒睡眠的进程让其完成 down 操作。

管程：使用信号量机制实现的生产者消费者问题需要客户端代码做很多控制，而管程把控制的代码独立出来，不仅不容易出错，也使得客户端代码调用更容易。管程有一个重要特性：在一个时刻只能有一个进程使用管程。进程在无法继续执行的时候不能一直占用管程，否者其它进程永远不能使用管程。

# 三、死锁

# 四、内存管理

# 五、设备管理