操作系统作业

1. **为什么要进行进程通信？进程通信的基本模式有哪些？**

显然，进程之间需要通信以实现数据传递，进而实现任务的协作完成。能影响其他进程或被其他进程影响的进程是协作进程。协作进程之间需要一种进程间通信机制(IPC)来使它们能够交换信息。IPC 的基本模式有共享内存与消息传递。

2. **那么共享内存与消息传递的具体实现方式是怎样的？**

首先通过生产者-消费者问题来解释共享内存模型。生产者产生信息供消费者消费，两者之间有一个类似于队列的缓冲区。生产者向内写入，消费者从中读出。在实现中，缓冲区需要有一个标志位以判断它是否为空。如果缓冲区为空，那么必将有一方出现等待。

对于消息传递模型，两个进程之间需要建立连接。连接有两种：直接通信与间接通信。在直接通信中，进程通过接收者的进程名来建立连接。这种实现方式比较简单，但是一旦某一方更改了进程名，那么所有的旧名称的引用都得被改过来。间接通信采用邮箱来收发信息。一个邮箱可以被多个进程共用，一个进程也可以使用多个信箱。这种方式的问题在于发送者和接受者容易出现混乱。

3. **客户机-服务器系统的通信有哪些具体实现方式呢？**

客户机-服务器系统是十分常见的系统。共享内存和消息传递可以用于此类系统。除此之外，考虑到这种系统的特点，还有其他的通信方式可以使用。

第一种是端口。端口可以定义为一个通信终端。一对进程提供一对端口，端口通过 IP 地址或者端口号进行区别。服务器通过端口对客户端进行监听，以此传输数据。第二种是远程过程调用。这种通信方式允许客户机调用远程主机上的过程。第三种是远程方法调用，这种通信方式是基于面向对象的，它允许调用远程对象的方法。

4. **什么是进程同步？我们为什么需要进行进程同步？**

如上所述，协作进程可能共享资源。如果多个进程同时使用共享资源，会产生冲突。这时需要安排进程的执行顺序，避免冲突，同时保证进程高效运行。这就是同步的意义。

5. **有哪几种实现进程同步的方式？**

每一个进程有一个代码段称为临界区，没有两个进程可以同时在临界区内执行。内核在实现调度的时候可以有两种调度方式：抢占和不允许抢占。不允许抢占是指操作系统规定在某一进程处于内核态时，任何其他进程不能发出中断指令使其暂停。允许抢占与之相反。允许抢占虽然使开销变大，但是可以使响应更为及时。

6. **那么实际中应该如何解决临界区问题呢？**

一种经典解决方案是 Peterson 算法。Peterson 算法是一种“谦让”的算法。如果某一进程想要进入临界区，那么就检查另一个进程是否也想进入临界区，即把对方的标志位置为 1。只有在另一进程不需要进入临界区，或者己方标志位为 1 时才可以进入临界区。这种方法可以有效的解决冲突，但是不足在于等待会浪费 CPU 时间，而且低优先级的进程可能会把高优先级的进程阻塞掉。

还有一种可以硬件实现的方式是使用互斥锁。当一个进程进入临界区时，先检查锁是否可用。如果可用，则获取锁，进入临界区，退出临界区时把锁交还。如果锁不可用，就等待。这种方法的问题在于如果一个进程没有把锁获取完就被切换，就会出现问题。因此，应当原子地进行锁的获取与释放，从而避免打断。互斥锁在多核场景使用较多。

虽然互斥锁是一种较好的解决方案，但是对程序员而言使用较为复杂。因此，临界区问题的最终解决方案实际是信号量。信号量类似于生活中的标志旗，用以调度进程的运行。信号量由两种原子的操作来控制：down 和 up。操作系统拥有一个初始信号量的值，每当一个进程使用资源，那么操作系统就会把信号量减一，并把资源分配出去。如果信号量为 0，那么后来的进程就得等待。

信号量会引起的问题是死锁。假如两个进程同时无限期地等待一个事件，但是事件只能由两个进程之一完成（比如互相等待对方结束运行），就会出现死锁。有很多方案可以解决死锁问题，但是最常见的解决方案就是不解决。这是因为一般来说死锁发生概率并不大，只需要用户重启进程即可，并且处理死锁需要很大的开销。

7. 有哪些经典同步问题？应该如何解决？

接下来介绍一些经典同步问题的例子。这些例子有助于理解信号量的实际应用。

第一个范例是哲学家进餐问题。受篇幅限制，在此不介绍问题的具体内容。其解决方案如下：首先哲学家拿起左侧的筷子，然后检查右侧的筷子是否占用。如果右侧的筷子被占用，就放下左侧的筷子。否则拿起右侧筷子开始进食，吃完后放下两只筷子。此外，每一次放下筷子的时候都要通知两侧的哲学家。如果他们想吃饭，就把筷子分给他们。

第二个范例是读者-写者问题。两个进程可以同时访问一个共享资源。但是读的时候不允许写，写的时候也不允许其他进程写。实现方法是：第一个读者开始读的时候 down,最后一个读者离开的时候 up。显然，这种方案里读的优先权比写高。

8. 进程调度是什么？在操作系统看来，进程有哪些状态？

众所周知，进程在执行过程中需要使用 CPU 进行运算，也要等待 I/O 设备。此外，操作系统为了实现多进程多任务运行，也需要一种合适高效的调度策略。

为实现调度，每一个进程在操作系统内部都有一个特定的状态与之对应。状态有新（刚刚创建），就绪，运行，以及等待。

9. 进程调度有哪些分类？有哪些算法？

进程调度分为抢占型调度与非抢占型调度。非抢占型调度指进程在运行中不能被打断。这种方式确实可以提高 CPU 效率，但是不能实现多任务，因此很早就不使用。现在最常使用的是抢占型调度。如果某一进程被调度器选中，那么它就不会停止。但是一旦它等待 I/O，就会被踢出 CPU 进行等待。

抢占型调度的最大挑战就是如何平衡公平性、策略执行与性能。策略的执行与公平性和性能有很大冲突。因此挑选一个合适的算法尤为重要。下面介绍一些经典算法。

第一种是先到先服务。这种算法严格按照请求顺序进行调度。它的缺点很明显：如果前面的进程占用大量时间，那么后面的进程就会大量积压。

第二种是短作业优先算法。这种算法在一个进程执行完后会挑选队列里最短的进程作业运行。这种算法的缺点也很明显：耗时长的进程可能会饿死。

第三种是轮转算法。这种算法工作模式类似于循环队列。它把时间分为一定长度的时间片，每过一个时间片就进行一次进程切换。这种方案需要选择合适的时间片长度。如果太短，切换过于频繁，资源消耗就会很大；如果太长，又和没有分片一样。一般来说，时间片的长度大概是 10-100ms。

第四种方法是优先级加不同等级进程队列的调度方案。这种方法参考了上述的三种方法，是实际使用的方法。在这种算法中，进程被赋予一个优先级。优先级高的进程采取先到先服务，较低的采用短作业优先，更低的采用轮转算法。轮转算法的时间片的长度也可以随优先级变化。

10. 优先级算法有哪些具体应用？

在多核系统中，有两种实践方案：非对称多处理与对称多处理。非对称多处理方法中 ，一个主处理器负责调度和处理 I/O，其余处理器只负责执行用户代码。而在对称多处理方法中，每一个处理器都有自己的就绪进程队列。

在实时系统中，系统的时效性尤为重要。这时要考虑进程的截止时间。有一种很好的解决方案就是检查进程的截止时间，哪一个进程的截止时间早就先执行哪一个。此外，很多情况下没有一种调度方案可以满足所有需要。

11. 总结

进程的通信、同步、调度是操作系统的重要功能。理解这三种功能的实现有助于我们编写性能更佳的程序。