面试宝典 - 其他

操作系统

操作系统概述

操作系统分类

操作系统分为：批处理操作系统（单道、多道）、分时操作系统（Unix）、实时操作系统（MsgOS）、网络操作系统、分布式操作系统、微机操作系统（Linux、Windows、IOS 等）、嵌入式操作系统（Android、IOS）。

操作系统的 4 个特性

1. 并发性

   区别于并行性，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. 共享性

   一般的共享是指某种资源可以被大家使用，在 OS 下的资源共享称为资源复用，具体含义是：系统中的资源可供内存中多个并发的执行的进程共同使用。

   系统中的资源可以被内存中多个并发执行的进线程共同使用。

   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，在同一时刻只允许一个进程访问，需要用同步机制来实现互斥访问。

3. 虚拟性

   通过某种技术将一个物理实体变为若干个逻辑上的对应物的功能即为虚拟，也就是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（时间）分复用技术（如分时系统）和空（空间）分复用技术（如虚拟内存）。

   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，让每个进程轮流占用处理器，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，每道程序完成的时间都是不可预知的，进程是以不可知的速度向前推进。

操作系统的基本功能

操作系统的功能有：进程管理、文件管理、存储管理、设备管理、作业管理。

进程管理

也称处理机管理，实质上是对处理机执行时间进行管理，采用多道程序等技术将 CPU 的时间真正合理地分配给每个任务。

主要包括进程管理、进程同步、进程通信和进程调度。

文件管理

又称信息管理，主要包括文件存储空间管理、目录管理、文件读写管理和存取管理。

内存管理

也称为存储管理，是对主存储器空间的管理。

主要包括存储分配与回收、存储保护与共享、地址映射（变换）和主存扩充。

设备管理

实质上是对硬件设备进行管理，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。

其中包括缓冲管理、输入输出设备的分配、启动、完成和回收、设备处理、虚拟设备。

作业管理

包括人物、人机交互和用户界面管理等。

进程与线程

进程

进程是具有一定独立功能的程序关于某个数据集合上地一次运行活动，是系统进行资源分配和调度的一个独立基本单位。

进程的特征

1. 结构特征：程序段、相关数据段和 PCB 三部分构成进程实体。
2. 动态性：进程实体的一次执行过程，具有生命期，而程序是有序指令集合，是静态的。
3. 并发性：多个进程同时存于内存，在一段时间内同时运行。
4. 独立性：进程实体是一个能独立运行、独立分配资源和独立接收调度的基本单位。
5. 异步性：进程按各自独立的、不可预知的速度向前推进。

多进程的组织形式包括下面 3 个关键部分

1. PCB（Process Control Block）：进程控制块，用来记录进程信息的数据结构（管理进程的核心，包含了 PID 等进程的所有关键信息）。

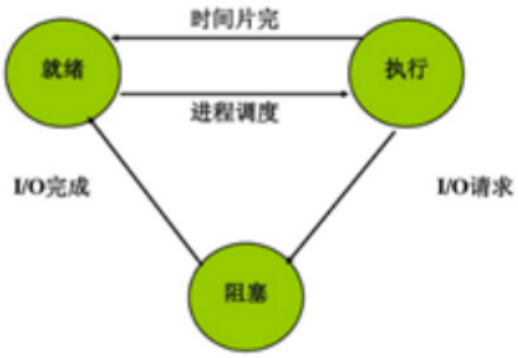
   描述进程的基本信息和运行状态，所谓的创建进程和撤销进程，都是指对 PCB 的操作。

   为了便于系统描述和管理进程，在 OS 的核心为每个进程专门定义了一个数据结构，进程控制块 PCB。PCB 是进程的唯一标志。

2. 进程的状态：
   - （1）就绪状态
   - （2）执行状态
   - （3）阻塞状态（多线程时也是这些状态）
3. 队列：就绪队列，等待（阻塞）队列。

进程的状态

三态模型（左图）、五态模型（右图）



- 就绪状态（ready）：等待被调度。
- 运行状态（running）
- 阻塞状态（waiting）：等待资源。

处于就绪状态的进程，在调度程序为之分配了处理机之后便开始执行，就绪 -> 执行。

正在执行的进程如果因为分配它的时间片已经用完，而被剥夺处理器，执行 -> 就绪。

如果因为某种原因致使当前的进程执行受阻，使之不能执行。执行 -> 阻塞。

应该注意以下内容：

- 只有就绪态和运行态可以互相转换，其它的都是单向转换。就绪状态的进程通过调度算法从而获得 CPU 时间，转为运行状态；而运行状态的进程，在分配给它的 CPU 时间片用完之后就会转为就绪状态，等待下一次调度。
- 阻塞状态是缺少需要的资源从而由运行状态转换而来，但是该资源不包括 CPU 时间，减少 CPU 时间会从运行态转换为就绪态。

线程

线程是进程的一个实体，是 CPU 独立调度和分派的基本单位，它是比进程更小的能独立运行的基本单位。线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源（如程序计数器、一组寄存器和栈），但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源，也有就绪、运行、阻塞三态。

一个进程中可以有多个线程，它们共享进程资源。

一个线程可以创建和撤销另一个线程，同一个进程中的多个线程之间可以并发执行。相对进程而言，线程是一个更加接近于执行体的概念，它可以与同进程中的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行序列。

线程有自己的 TCB （thread control block 线程控制块），只负责这条流程的信息，包括 PC 程序计数器，SP 栈、State 状态、寄存器、线程 id。

线程分类

线程有内核级线程和用户级线程，一般说的都是用户级线程，内核级线程由内核管理。

1. 只有内核级线程才能发挥多核性能，因为内核级线程共用一套 MMU（即内存映射表），统一分配核 1 核 2 （即有多个 CPU，可以一个 CPU 执行一个内核级线程），进程无法发挥多核性能，因为进程切换都得切 MMU。
2. 为什么需要内核级线程？如果只有用户级线程，在内核中只能看到进程，所以当用户级线程中一个线程进行 IO 读写阻塞时，内核会将该线程所在的进程直接切换。例如当用浏览器打开网页，这个进程中有下载数据线程，有显示数据线程，当数据下载读写阻塞时，内核直接切到 qq（这些切换是指在 CPU 上运行的程序）。

进程与线程

区别

1. 拥有资源

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

   进程 = 资源（包括寄存器值，PCB，内存映射表）+ TCB（栈结构）。

2. 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。

   线程 = TCB（栈结构）。

3. 系统开销

   由于创建或撤销进程时，系统都要为之分配或回收资源，如内存空间、I/O 设备等，所付出的开销远大于创建或撤销线程时的开销。类似地，在进行进程切换时，涉及当前执行进程 CPU 环境的保护及新调度进程 CPU 环境的设置，而线程切换时只需保存和设置少量寄存器的内容，开销很小。

   线程的切换只是切换 PC，切换了 TCB（栈结构）。

   进程的切换不仅要切换 PC，还包括切换资源，即切换内存映射表。

4. 通信方面

进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程的资源是共享的。

进程间的资源是分隔独立的，内存映射表不同，占用物理内存地址是分隔的。

线程间可以通过直接读写同一进程中的数据进行通信，但是进程通信需要借助 IPC。

5. 执行过程

   线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

6. 其他

   一个程序至少有一个进程，一个进程至少有一个线程。

   线程的划分尺度小于进程，使得多线程程序的并发性高。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，用来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

优缺点

线程执行开销小，但不利于资源的管理和保护；而进程正相反。同时，线程适合于在 SMP 机器上运行，而进程则可以跨机器迁移。

进程同步

经典的进程同步问题：生产者-消费者问题；哲学家进餐问题；读者-写者问题。

同步的解决方案：管程、信号量。

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：多个进程因为合作产生的直接制约关系，使得进程有一定的先后执行关系。

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 Semaphore

信号量机制：即利用 PV 操作来对信号量进行处理。

信号量（Semaphore）的数据结构为一个值和一个指针，指针指向等待该信号量的下一个进程。信号量的值与相应资源的使用情况有关。

当它的值大于 0 时，表示当前可用资源的数量。

当它的值小于 0 时，其绝对值表示等待使用该资源的进程个数。

注意，信号量的值仅能由 PV 操作来改变。

P 和 V 操作，也就是常见的 down 和 up 操作：

- P：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；
- V：对信号量执行 +1 操作，唤醒睡眠的进程让其完成 down 操作。

P和 V 操作需要被设计成原语，不可分割，通常的做法是在执行这些操作的时候屏蔽中断。

一般来说，信号量 S >= 0 时，S 表示可用资源的数量。执行一次 P 操作意味着请求分配一个单位资源，因此 S 的值减 1；当 S < 0 时，表示已经没有可用资源，请求者必须等待别的进程释放该类资源，它才能运行下去。而执行一个 V 操作意味着释放一个单位资源，因此 S 的值加 1；若 S <= 0，表示有某些进程正在等待该资源，因此要唤醒一个等待状态的进程，使之运行下去。

如果信号量的取值只能为 0 或者 1，那么就成为了互斥量（Mutex），0 表示临界区已经加锁，1 表示临界区解锁。

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，否则其他进程永远不能使用管程。

管程引入了条件变量以及相关的操作：wait() 和 signal() 来实现同步操作。对条件变量执行 wait() 操作会导致调用进程阻塞，把管程让出来给另一个进程持有。signal() 操作用于唤醒被阻塞的进程。

进程通信

进程通信是一种手段，而进程同步是一种目的。也就是说，为了能够达到进程同步的目的，需要让进程进行通信，传输一些进程同步所需要的信息。

由于多个进程可以并发执行，所以进程间必然存在资源共享和相互合作的问题。进程通信是指各个进程交换信息的过程。

同步是合作进程间直接制约问题，互斥是申请临界资源进程间的间接制约问题。

临界资源（Critical Resource,CR）：在同一时间只能供一个进程使用的资源，例如：打印机、磁带机等硬件资源。

临界区：每个进程中访问临界资源的那段代码。

临界区管理 4 条原则：

1. 有空即进。
2. 无空则等。
3. 有限等待：要求访问临界区的进程，保证有限时间内进入临界区，避免死等。
4. 让权等待：进程不能进入临界区时，应立即释放处理机，避免忙等。

进程同步与进程通信很容易混淆，它们的区别在于：

- 进程同步：控制多个进程按一定顺序执行。
- 进程通信：进程间传输信息。

进程间通信的手段有：管道、FIFO（命名管道）、消息队列、信号量、共享存储、套接字。

管道

管道是通过调用 pipe 函数创建的，fd[0] 用于读，fd[1] 用于写。

它具有以下限制：

- 只支持半双工通信（单向交替传输）；
- 只能在父子进程或者兄弟进程中使用。

FIFO

也称为命名管道，去除了管道只能在父子进程中使用的限制。

FIFO 常用于客户-服务器应用程序中，FIFO 用作汇聚点，在客户进程和服务器进程之间传递数据。

消息队列

相比于 FIFO，消息队列具有以下有点：

1. 消息队列可以独立于读写进程存在，从而避免了 FIFO 中同步管道的打开和关闭时可能产生的困难；
2. 避免了 FIFO 的同步阻塞问题，不需要进程自己提供同步方法；
3. 读进程可以根据消息类型有选择地接收消息，而不像 FIFO 那样只能默认地接收。

信号量

它是一个计数器，用于为多个进程提供对共享数据对象的访问。

共享存储

允许多个进程共享一个给定的存储区。因为数据不需要再进程之间复制，所以这是最快的一种 IPC。

需要使用信号量来同步对共享存储的访问。

多个进程可以将同一个文件映射到它们的地址空间从而实现共享内存。另外 XSI 共享内存不是使用文件，而是使用内存的匿名段。

套接字

与其他通信机制不同的是，它可用于不用机器间的进程通信。

池

所谓池的概念，一般是指应用提前向内核批量申请资源，用于接下来的使用和回收，减少资源的初始化和销毁次数等开销，以达到提高系统性能的目标。

内存池：真正使用前申请一片内存区域，有新需求时取出其中一部分使用，不够用时再重新申请新内存。

进程池：应用预先创建一组子进程，所有子进程运行相同代码，拥有相同属性，比如 PGID 和优先级等。

常见两种工作方式：

1. 主进程通过随机或 round robin 算法来选择子进程作为新任务的服务进程；

2. 通过一个共享队列来进行同步，所有子进程从该队列中获取任务，不过同时只能有一个子进程能成功获得新任务处理权。

线程池：主要应用于任务小而多，处理时间短的场景，比如简单网页请求等。

死锁

死锁概念

两个以上的进程互相要求双方释放已经占用的资源导致无法继续运行下去的现象。

在两个或多个并发进程中，如果每个进程持有某种资源而又等待别的进程释放它或它们现在保持着的资源，在未改变这种状态之前都不能向前推进，称这一组进程产生了死锁。通俗地讲，就是两个或多个进程被无限期地阻塞、相互等待地一种状态。

例如：一个系统有一台扫描仪 R1，一台刻录机 R2，有两个进程 P1、P2，他们都准备将扫描的文档刻录到 CD 上，P1 先请求 R1 成功，P2 先请求 R2 并成功，后来，P1 又请求 R2，但却因为已经分配而阻塞，P2 请求 R1，也因分配而阻塞，此时，双方都被阻塞，都希望双方释放自己所需的资源，但又谁都不能得到自己所需的资源而继续进程，从而一直占用自己所占的资源，就形成死锁。

可抢占性资源和不可抢占性资源

可抢占性资源：某进程获得这类资源后，该资源可以在被其他进程或系统抢占。

不可抢占性资源：一旦系统将资源分配给一个进程以后，就不能把它强行收回，只能等它用完自行释放。

产生死锁的原因

1. 竞争不可抢占性资源。
2. 进程运行推进的顺序不合适。
3. 竞争可消耗资源。

如果系统资源充足，进程的资源请求都能够得到满足，死锁出现的可能性就很低，否则就会因抢夺优先的资源而陷入死锁。其次，进程运行推进顺序与速度不同，也可能产生死锁。

产生死锁的四个必要条件

产生死锁的四个必要条件：互斥条件、请求与保持条件、不可剥夺条件、循环等待条件。

互斥条件

一个资源一次只能被一个进程使用。

每个资源要么已经分配给了一个进程，要么就是可用的。

请求与保持条件

一个进程因请求资源而阻塞时，对已获得的资源保持不放。

已经得到了某个资源的进程可以再请求新的资源。

不可剥夺条件

进程已获取的资源，在未使用完之前，不能强行剥夺。

已经分配给一个进程的资源不能强制性地被抢占，它只能被占用它的进程显式地释放。

循环等待条件

若干进程之间形成一种头尾相连的循环等待资源关系。

有两个或者两个以上的进程组成一条环路，该环路中的每个进程都在等待下一个进程所占用的资源。

这四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

处理方法

解决死锁的 4 种处理策略：鸵鸟策略（即不理睬策略）、预防策略、避免策略、检测与解除策略。

1. 鸵鸟策略：Windows、Linux 个人版都不做死锁处理，直接忽略，大不了重启就好了，小概率事件，代价可以接受。
2. 预防死锁：破坏产生死锁的 4 个必要条件中的一个或者多个；实现起来比较简单，但是如果限制过于严格会降低系统资源利用率以及吞吐量。

   死锁预防的方法有：

   1. 预先静态分配法：破坏不可剥夺条件。
   2. 资源有序分配法：将资源分类按顺序排列，保证不形成环路。

      破坏互斥条件：例如打印技术允许若干个进程同时输出，唯一真正请求物理打印机的进程是打印守护进程。

      破坏占有和等待条件：一种实现方式是规定所有进程在开始执行前请求所需要的全部资源。

      破坏不可抢占条件：进程可以抢占其他进程持有的资源。

      破坏环路等待：给资源同一编号，进程只能按编号顺序来请求资源。
3. 避免死锁：在资源的动态分配中，防止系统进入不安全状态（可能产生死锁的状态），如银行家算法。

   银行家算法可以对每个资源请求进行检测，确保安全，在程序运行时避免发生死锁。

   单个资源的银行家算法：一个小城镇的银行家，他向一群客户分别承诺了一定的贷款额度，算法要做的是判断对请求的满足是否会进入不安全状态，如果是，就拒绝请求；否则予以分配。
4. 检测死锁：允许系统运行过程中产生死锁，在死锁发生之后，采用一定的算法进行检测，并确定与死锁相关的资源和进程，采取相关方法清除检测到的死锁。实现难度大。
5. 解除死锁：与死锁检测配合，将系统从死锁中解脱出来（撤销进程或者剥夺资源）。对检测到的和死锁相关的进程以及资源，通过撤销或者挂起的方式，释放一些资源并将其分配给处于阻塞状态的进程，使其转变为就绪态。实现难度大。