第16章

操作系统

操作系统面试例题主要包括进程、线程、内存管理、垃圾回收，以及缓存等诸多方面。

16.1 进程

面试例题1： 试解释操作系统原理中的作业、进程、线程、管程各自的定义。[中国著名通讯公司Z2009年10月笔试题]

解析：操作系统面试问题。

答案：作业：用户在一次解题或一个事务处理过程中要求计算机系统所做工作的集合。它包括用户程序、所需要的数据及控制命令等。作业是由一系列有序的步骤组成的。

进程：一个程序在一个数据集合上的一次运行过程。所以一个程序在不同数据集合上运行，乃至一个程序在同样数据集合上的多次运行都是不同的进程。

线程：线程是进程中的一个实体，是被系统独立调度和执行的基本单位。

管程：管程实际上是定义了一个数据结构和在该数据结构上的能为并发进程所执行的一组操作，这组操作能同步进程和改变管程中的数据。

面试例题 2： 进程间的通信如何实现？[日本某著名家电/通信/IT 企业面试题]

答案：现在最常用的进程间通信的方式有信号、信号量、消息队列、共享内存。所谓进程通信，就是不同进程之间进行一些“接触”。这种接触有简单，也有复杂。机制不同，复杂度也不一样。通信是一个广义上的意义，不仅仅指传递一些 message。它们的使用

方法是基本相同的，所以只要掌握了一种使用方法，然后记住其他的使用方法就可以了。信号和信号量是不同的，它们虽然都可用来实现同步和互斥，但前者是使用信号处理器来进行的，后者是使用 P、V 操作来实现的。消息队列是比较高级的一种进程间通信方法，因为它真的可以在进程间传送 message，连传送一个“I seek you”都可以。

一个消息队列可以被多个进程所共享（IPC 就是在这个基础上进行的）；如果一个进程的消息太多，一个消息队列放不下，也可以用多于一个的消息队列（不过可能管理会比较复杂）。共享消息队列的进程所发送的消息中除了 message 本身外还有一个标志，这个标志可以指明该消息将由哪个进程或者是哪类进程接受。每一个共享消息队列的进程针对这个队列也有自己的标志，可以用来声明自己的身份。

面试例题 3: 在 Windows 编程中互斥器（mutex）的作用和临界区（critical section）类似，请说一下二者间的主要区别。[中国台湾某著名杀毒软件公司 2005 年面试题]

解析: 多线程编程问题。

答案: 两者的区别是 mutex 可以用于进程之间互斥，critical section 是线程之间的互斥。

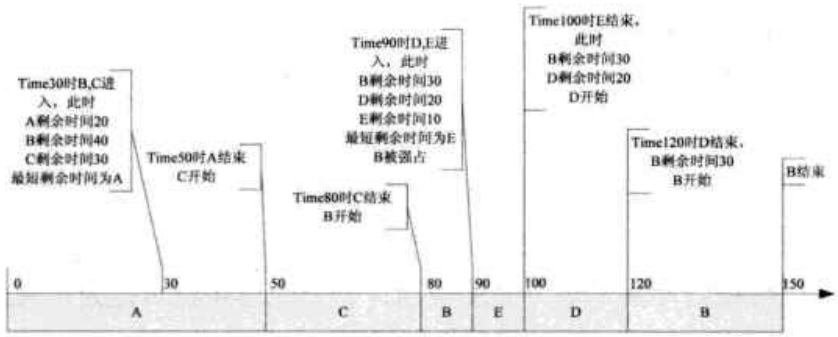
面试例题 4: At time 0, process A has arrived in the system, in that order; at time 30, both progress B and C have just arrived; at time 90, both progress D and E have also arrived.The quantum or timeslice is 10 units.（在 0 时刻，进程 A 进入系统，按照这个顺序，在 30 时刻，进程 B 和进程 C 也抵达；在 90 时刻，进程 D 和进程 E 也抵达。一个时间片是 10 个单元）

Process A requires 50 units of time in the CPU; Process B requires 40 units of time in the CPU; Process C requires 30 units of time in the CPU; Process D requires 20 units of time in the CPU; Process E requires 10 units of time in the CPU。

（进程 A 需要占用 CPU50 个单元；进程 B 需要占用 CPU40 个单元；进程 C 需要占用 CPU30 个单元；进程 D 需要占用 CPU20 个单元；进程 E 需要占用 CPU10 个单元。）

Which of the process will be the LAST to complete, if scheduling policy is preemptive SJF (Short Job First)?（如果按照短作业优先级的方法，哪个进程最后结束？）[美国著名软件公司 M2009 年 10 月笔试题]

解析: 牢记“短作业优先” = “最短剩余时间作业优先”。本题考的是可剥夺式处理机的调度问题。时间图如下所示。



答案: 如果按短作业优先级的方法，进程 B 最后结束。

扩展知识:

读者试着扩展思维，想想如果本题是不可剥夺式处理机的调度方法，哪个进程最后结束。

面试例题 5: For a multiple-processing OS, the way to deal with dead-lock is whether to prevent it from happening, or break it happens. Which of the following approaches belong to the way to prevent dead-lock from happening?(M)（多选：在多重处理系统中，处理死锁的办法有两种：一是防止其发生；二是发生后进行处理。下面的办法中属于防止其发生的是哪一个？）[中国台湾著名杀毒软件公司 Q2009 年 1 月笔试题]

A. Destroy mutual condition（破坏互斥条件）

B. Destroy non-preemptive condition （破坏不可剥夺条件）

C. Destroy iterative waiting condition （破坏循环等待条件）

D. To kill one process which involves in dead-lock（杀死某个激活死锁的进程）

解析: 所谓 deadlocks（死锁）是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁。

产生死锁的 4 个必要条件:

- 互斥条件: 一个资源每次只能被一个进程使用。
- 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。
- 不剥夺条件: 进程已获得的资源，在未使用完之前，不能强行剥夺。
- 循环等待条件: 若干进程之间形成一种头尾相接的循环等待资源关系。

这 4 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。

死锁的解除与预防方法如下:

理解了死锁的原因，尤其是产生死锁的 4 个必要条件，就可以最大可能地避免、预防和解除死锁。所以，在系统设计、进程调度等方面注意如何不让这 4 个必要条件成立，如何确定资源的合理分配算法，避免进程永久占据系统资源。此外，也要防止进程在处于等待状态的情况下占用资源，在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统可能发生死锁，则不予分配，否则予以分配 。因此，对资源的分配要给予合理的规划。

根据产生死锁的 4 个必要条件，只要使其中之一不能成立，死锁就不会出现。为此，可以采取下列 3 种预防措施:

- 采用资源静态分配策略，破坏“部分分配”条件。
- 允许进程剥夺使用其他进程占有的资源，从而破坏“不可剥夺”条件。
- 采用资源有序分配法，破坏“环路”条件。

这里注意一点：互斥条件无法被破坏。

死锁的避免不严格地限制死锁的必要条件的存在，而是系统在系统运行过程中小心地避免死锁的最终发生。避免死锁算法中最有代表性的算法是 Dijkstra E.W 于 1968 年提出的银行家算法，该算法需要检查申请者对资源的最大需求量，如果系统现存的各类资源可以满足申请者的请求，就满足申请者的请求。这样申请者就可很快完成其计算，然后释放它占用的资源，从而保证了系统中的所有进程都能完成，所以可避免死锁的发生。

在本题中 选项 A 破坏互斥条件是无法做到的，B、C 正确，D 选项属于死锁事后处理操作，与题意不符。

答案: B，C。

16.2 线程

面试例题 1: 请描述进程和线程的差别。[美国某著名软件公司 2005 年面试题]

答案: 进程是程序的一次执行。线程可以理解为进程中执行的一段程序片段。在一个多任务环境中下面的概念可以帮助我们理解两者间的差别。

进程间是独立的，这表现在内存空间、上下文环境上；线程运行在进程空间内。 一般来讲（不使用特殊技术），进程无法突破进程边界存取其他进程内的存储空间；而线程由于处于进程空间内，所以同一进程所产生的线程共享同一内存空间。

同一进程中的两段代码不能够同时执行，除非引入线程。

线程是属于进程的，当进程退出时该进程所产生的线程都会被强制退出并清除。线程占用的资源要少于进程所占用的资源。进程和线程都可以有优先级。

面试例题 2：下面哪个选项不是 PE 文件？

A．EXE　B．DLL　　C．COM　　D．DOC

解析：PE 文件被称为可移植的执行体是 Portable Execute 的全称，常见的 EXE、DLL、OCX、SYS、COM 都是 PE 文件，PE 文件是微软 Windows 操作系统上的程序文件（可能是间接被执行，如 DLL）

答案：D

面试例题 3：Windows 将遵循下面的哪种搜索来定位 DLL？

1 进程的当前工作目录

2 包含 EXE 文件的目录

3 列在 Path 环境变量中的一系列目录

4 Windows 系统目录

5 Windows 目录

A．12453　B．12543　C．21453　D．21345

解析： Windows 平台的大多数程序都使用各种动态链接库 （DLL） 来避免重复实现功能。操作系统为每个程序加载若干个 DLL，具体由程序的类型决定。当程序不指定 DLL 的绝对位置时，将使用默认的搜索顺序来找到它。默认情况下，操作系统所使用的搜索顺序为：（1）内存；（2）KnownDLLs；（3）清单与 .local；（4）应用程序目录；（5）当前工作目录；（6）系统目录；（7）路径变量。

答案：A

面试例题 4：DLL 文件是什么？它有几种调用方式？

答案：DLL 文件（Dynamic Linkable Library 即动态链接库文件），是一种不能单独运行的文件，它允许程序共享执行特殊任务所必需的代码和其他资源。比较大的应用程序都由很多模块组成，这些模块分别完成相对独立的功能，它们彼此协作来完成整个软件系统的工作。可能存在一些模块的功能较为通用，在构造其他软件系统时仍会被使用。在构造软件系统时，如果将所有模块的源代码都静态编译到整个应用程序 EXE 文件中，会产生一些问题：一个

是增加了应用程序的大小，会占用更多的磁盘空间，程序运行时也会消耗较大的内存空间，造成系统资源的浪费；另一个是，在编写大的 EXE 程序时，在每次修改重建时都必须调整编译所有源代码，增加了编译过程的复杂性，也不利于阶段性的单元测试。

一般来说，DLL 是一种磁盘文件，以.dll、.DRV、.FON、.SYS 和许多以 .EXE 为扩展名的系统文件都可以是 DLL。它由全局数据、服务函数和资源组成，在运行时被系统加载到调用进程的虚拟空间中，成为调用进程的一部分。如果与其他 DLL 之间没有冲突，该文件通常映射到进程虚拟空间的同一地址上。DLL 模块中包含各种导出函数，用于向外界提供服务。DLL 可以有自己的数据段，但没有自己的堆栈，使用与调用它的应用程序是相同的堆栈模式；一个 DLL 在内存中只有一个实例；DLL 实现了代码封装性；DLL 的编制与具体的编程语言及编译器无关。

调用方式有以下两种。

- 静态调用方式：由编译系统完成对 DLL 的加载和应用程序结束时 DLL 卸载的编码（如还有其他程序使用该 DLL，则 Windows 对 DLL 的应用记录减 1，直到所有相关程序都结束对该 DLL 的使用时才释放它，此方式简单实用，但不够灵活，只能满足一般要求。
- 动态调用方式：是由编程者用 API 函数加载和卸载 DLL 来达到调用 DLL 的目的，使用上较复杂，但能更加有效地使用内存，是编制大型应用程序时的重要方式。

正因为 DLL 有占用内存小，好编辑等特点，所以有很多电脑病毒都是 DLL 格式文件。但不能单独运行。动态链接库通常都不能直接运行，也不能接收消息。它们是一些独立的文件，其中包含能被可执行程序或其他 DLL 调用来完成某项工作的函数。只有在其他模块调用动态链接库中的函数时，它才发挥作用。

16.3 内存管理

面试例题 1： 垃圾回收的优点和原理是什么？并考虑两种回收机制。

答案： Java 语言中一个显著的特点就是引入了垃圾回收机制，使 C++程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。由于有垃圾回收机制，Java 中的对象不再有“作用域”的概念，只有对象的引用才有“作用域”。垃圾回收可以有效地防止内存泄露，有效地使用可以使用的内存。垃圾回收器通常作为一个单独的低级别的线程运行，在不可预知的情况下对内存堆中已经死亡的或者长时间没有使用

的对象进行清除和回收，程序员不能实时地调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收、标记垃圾回收和增量垃圾回收 3 种。

面试例题 2: What is “cache” in CPU and “cache” in OS?（CPU 中的缓存和操作系统中的缓存分别是什么？）[美国某著名计算机嵌入式公司面试题]

答案:

1．快表——Cache 在 OS 中运用的典型范例

在操作系统中，为提高系统的存取速度，在地址映射机制中增加一个小容量的联想寄存器（相联存储器），即快表，用来存放当前访问最频繁的少数活动页面的页号。当某用户需要存取数据时，根据数据所在的逻辑页号在快表中找到其对应的内存块号，再联系页内地址，形成物理地址。如果在快表中没有相应的逻辑页号，则地址映射仍可以通过内存中的页表进行，得到空闲块号后须将该块号填入快表的空闲块中。如果快表中没有空闲块，则根据淘汰算法淘汰某一行，再填入新的页号和块号。

快表查找内存块的物理地址消耗的时间大大降低了，使得系统效率得到了极大的提高。

例如，Linux 使用页面 Cache 的目的是加快对磁盘上文件的访问。内存映射文件以每次一页的方式读出并将这些页面存储在页面 Cache 中。

2．高速缓冲存储器（Cache）——Cache 在 CPU 中运用的典型范例

CPU 的执行速度越来越快，系统架构越来越先进，而主存的结构和存取速度改进则较慢，因此，高速缓存技术将越来越重要。

高速缓冲存储器（Cache）是位于 CPU 与内存之间的临时存储器，它的容量比内存小但交换速度快。在 Cache 中的数据是内存中的一小部分，但这一小部分是短时间内 CPU 即将访问的。当 CPU 调用大量数据时，就可避开内存直接从 Cache 中调用，从而加快读取速度。由此可见，在 CPU 中加入 Cache 是一种高效的解决方案，这样整个内存储器（Cache+内存）就变成了既有 Cache 的高速度又有内存的大容量的存储系统了。Cache 对 CPU 性能的影响很大，这主要是由 CPU 的数据交换顺序和 CPU 与 Cache 间的带宽引起的。