# 操作系统

**进程是什么**

进程是具有一定独立功能的程序，它是系统进行资源分配和调度的一个独立单位，重点在系统调度和独立的单位，也就是说进程是可以独立运行的一段程序。

**线程是什么**

线程是进程的一个实体，是CPU调度和分配的基本单位，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，在运行时，只是暂用一些计数器，寄存器，和栈。

**进程与线程的关系**

* 一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程（通常说的主线程）
* 资源分配给进程，同一进程的所有线程共享该进程的所有资源
* 线程在执行过程中，需要协作同步，不同进程的线程间要利用消息通信的方法实现同步
* 处理机分配给线程，即真正在处理机上运行的是线程
* 线程是指进程内的一个执行单元，也是进程内的可调度实体

**从4个角度来分析进程与线程之间的区别**

* 调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位
* 并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可以并发执行
* 拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但是可以访问隶属于进程的资源
* 系统开销：创建或撤销进程时，系统都要为之分配或回收系统资源，如内存空间，I/O设备等，OS所付出的开销显著大于在创建或撤销线程时的开销，进程切换的开销也远大于线程切换的开销

**进程有几种状态**

* 就绪状态：进程已经获得除处理机以外的所有资源，等待分配处理机资源
* 运行状态：占用处理机资源运行，处于此状态的进程数小于等于CPU数目
* 阻塞状态：进程等待某种条件，在条件满足之前无法运行

**线程有几种状态**

* \*\*执行状态：\*\*表示线程已经获得处理机而正在运行
* \*\*就绪状态：\*\*指线程已经具备了各种执行条件，只需再获得CPU便可以立即执行难
* \*\*阻塞状态：\*\*线程在执行过程中因某事受阻而处于暂停状态

**操作系统进程调度有哪几种**

* \*\*非抢占式方式：\*\*采用这种调度方式时，一旦把处理机分配给某个进程后，就一直让它运行下去，决不会因为时钟中断或任何其它原因去抢占当前正在运行进程的处理机，直至该进程完成，或发生某种事件而被阻塞时，才把处理机分配给其他进程
* \*\*抢占式方式：\*\*这种调度方式允许调度程序根据某种原则，去暂停某个正在执行的进程，将已经分配给该进程的处理机重新分配给另一进程，在现代OS中广泛采用抢占方式。

**进程为什么需要同步**

进程同步机制的主要任务是对多个相关进程在执行次序上进行协调，使得并发执行的各个进程之间能按照一定的规则或时序共享系统资源，并能很好的相互合作，从而使得程序的执行具有可再现性

**列举几种进程同步机制，并说明其优缺点**

* **硬件同步机制：**
* **信号量机制：**
* **管程机制：**

**进程之间通信的途径**

* **管道**：管道是一种半双工的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用，进程的亲缘关系通常是指父子进程关系
* **有名管道**：有名管道也是半双工的通信方式，但是它允许无亲缘关系的进程间的通信
* **信号量**：信号量是一个计数器，可以用来控制多个进程对共享资源的访问，它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源，因此，主要作为进程间以及同一进程内不同线程之间的同步手段
* **消息队列**：消息队列是由消息的链表，存放在内核中并由消息队列标识符标识，消息队列克服了信号传递信息少，管道只能承载无格式字节流以及缓冲区大小受限等缺点
* **信号**：信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生
* **共享内存**：共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问，共享内存是最快的IPC方式，它是针对其他进程通信方式运行效率低而专门设计的，它往往与其他通信机制，如信号量，配合使用，来实现进程间的同步和通信
* **套接字**：套接字可以用于不同的进程间的通信

**线程之间通信的途径**

* **锁机制**：包括互斥锁，条件变量，读写锁  
  互斥锁提供了以排他方式防止数据结构被并发修改的方法  
  条件变量可以以原子的方式阻塞进程，直到某个特定条件为真为止，对条件的测试是在互斥锁的保护下进行的，条件变量始终与互斥锁一起使用  
  读写锁允许多个线程同时读共享数据，而对写操作是互斥的
* **信号量机制**：包括无名线程信号量和命名线程信号量
* **信号机制**：类似进程间的信号处理  
  **注意：线程间的通信的目的主要是用于线程的同步，所以线程没有像进程通信中的用于数据交换的通信机制**

**死锁的定义**

如果一组进程中的每一个进程都在等待仅由该组进程中的其它进程才能引发的事件，那么该组进程是死锁的

**进程死锁的原因**

多个进程对资源的争夺，包括对不可抢占资源和对可消耗资源

**进程死锁的4个必要条件**

* \*\*互斥条件：\*\*进程对所分配到的资源进行排它性使用，即在一段时间内，其资源只能被一个进程占用，如果此时还有其它进程请求该资源，则请求进程只能等待，直至占有该资源的进程用完释放
* \*\*请求和保持条件：\*\*进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源已经被其它进程占有，此时请求进程被阻塞，但对自己已经获得的资源保持不放
* \*\*不可抢占条件：\*\*进程已经获得的资源在未使用完之前不能被抢占，只能在进程使用完时由自己释放
* \*\*循环等待条件：\*\*在发生死锁时，必然存在一个进程-资源循环链，即进程集合｛P0,P1,…,PN｝中的P0正在等待一个P1占用的资源，P1正在等待P2占用的资源，…，PN 正在等待P0占用的资源

**进程死锁的处理**

* \*\*预防死锁：\*\*通过设置某些限制条件，去破坏产生死锁的四个必要条件中的一种
* \*\*避免死锁：\*\*在资源的动态分配过程中，用某种方法防止系统进入不安全状态
* \*\*检测死锁：\*\*通过检测机构及时检测出死锁的发生，然后采取适当的措施，把进程从死锁中解脱出来
* \*\*解除死锁：\*\*当检测到系统中已经发生死锁时，采取相应的措施，将进程从死锁状态解脱出来，常用的方法是撤销一些进程，回收他们的资源，将它们分配给已经处于阻塞状态的进程，使其能够继续运行

**描述实时操作系统的基本特征**

实时操作系统是指操作系统工作时，其各种资源可以根据需要随时进行动态分配，由于各种资源可以进行动态分配，因此，其处理事务的能力较强  
换言之，在特定时间内完成特定的任务，具有实时性与可靠性

**操作系统中断与轮询的基本特点**

* \*\*轮询：\*\*对I/O设备的程序轮询的方式是早期的计算机系统对I/O设备的一种管理方式，它定时对各种设备轮流询问一遍有无处理要求。轮询占据了CPU相当一部分处理时间，因此，程序轮询是一种效率较低的方式，在现代计算机系统中已经很少应用
* \*\*中断：\*\*中断是指CPU在正常运行程序的过程中，由于发生的内部或外部的特定的事件，使得CPU中断正在运行的程序，而转到响应的服务程序去处理

**什么是临界区，如何解决冲突**

每个进程中访问临界资源的那段程序称为临界区，每次只准许一个进程进入临界区，进入后不允许其他进程进入  
解决冲突的方法：

* 如果有若干个进程要求进入空闲的临界区，一次仅允许一个进程进入
* 任何时候，处于临界区内的进程不可以多于一个，如果已经有进程进入自己的临界区，则其他所有试图进入临界区的进程必须等待
* 进入临界区内的进程要在有限时间内退出，以便其它进程能及时进入自己的临界区
* 如果进程不能进入自己的临界区，则应该让出CPU，避免进程出现“忙碌”现象。

**windows下内存是如何管理的**

windows操纵内存可以分为两个层面，物理内存和虚拟内存

**嵌入式C语言面试题目**

**C语言中如何引用一个已经定义过的变量**

可以使用extern或者引用头文件的方式  
－ **extern**：  
－ **引用头文件**：  
注意：当定义写错时，注意两者在编译或者链接时产生错误的区别

**什么是预编译，何时需要预编译**

预编译是指程序执行前的一些预处理工作，主要指#表示的。  
何时需要预处理？

* 总是使用不经常改动的大型大码体
* 程序由多个模块组成，所有模块都使用一组标准的包含头文件和相同的编译选项

**C语言中关键字static的作用**

* **static全局变量**：全局变量的说明之前加上static就变成了静态的全局变量，全局变量本身就是静态存储，因此这两种在存储方式上没有区别。两者的区别在于非静态全局变量的作用域是整个源程序，也就是说当一个程序包含多个文件时，非静态的全局变量在多个文件中都是有效的，而静态全局变量则限制了其作用域，只能在定义该变量的源文件内有效
* **static局部变量**：存储方式变为静态存储
* **static函数**：将函数的作用域限制在当前文件

**C语言中关键字const的作用**

* 表示常量不可以修改
* 可以修饰函数的输入参数
* 修饰函数，防止更改  
  \*\*注意：\*\*在声明一个简单变量和数组时使用关键字const很简单，指针则会复杂一些，因为需要区分这两种情况：1. 指针本身成为const 2. 指针指向的值成为const  
  例如：

//第一种情况，pf指向一个常量浮点数值，但是pf本身的值是可以改变的，例如，它可以指向另外一个const值

const float \* pf;

//第二种情况，pt指向同一个地址，但是所指向的值是可以改变的

float \*const pt;

//第三种情况，ptr指向同一个位置，并且它所指向位置存储的值也不能改变

const float \* const ptr;

* 1
* 2
* 3
* 4
* 5
* 6

**总而言之，一个位于*左边任意位置的const使得数据成为常量，而一个位于*右边的const使得指针自身成为常量**

**C语言中关键字volatile的作用**

限定词volatile 告诉编译器该变量除了可被程序改变以外还可以被其他代理改变，典型的，它被用于硬件地址和其他并行运行的程序共享的数据。  
例如一个地址中可能保存着当前的时钟时间，不管程序做些什么，该地址的值都会随着时间而改变。  
又例如一个地址被用来接收来自其他计算机的信息。  
\*\*注意点：\*\*一个值可以同时是const和volatile，例如，硬件时钟一般设定为不能由程序改变，这一点使得它成为const，但它可以被程序以外的代理改变，这使得它成为volatile的，只需在声明中同时使用这两个限定词，顺序并不重要，如下所示：

volatile const int loc;

const volatile int \* ploc;

* 1
* 2

**全局变量可不可以定义在被多个.C文件包含的头文件中**

这个问题怎么说呢，就算是可以吧！这个全局变量可以被任何一个引用该头文件的c源文件中使用，但是只能有一个c源文件对该变量进行赋值，不然链接的时候会出现错误

**C语言的堆，栈，静态区的区别**

－ 程序的局部变量存储于栈中，全局变量存储于静态区，动态申请数据存储于堆中  
－ 一个由C编译的程序占用以下几个区：

1. **栈区**：由编译器自动分配与释放，存放函数的参数值，或者局部变量等，其操作方式类似于数据结构中的栈
2. **堆区**：由程序员分配与释放，若程序员不释放，则OS在结束时可能会释放，注意这里的堆是从编程语言的角度来说的（我曾经面试了一个学校，被问到**从编程语言的角度**说明堆和栈的区别，当时被问的一脸蒙蔽，呵呵呵），这个堆和数据结构中的堆是完全不一样的东西，至于为什么也叫堆，还请大佬们补充补充！
3. **全局区（静态区）**：全局变量和静态变量是存放在一块的，初始化的全局变量和静态变量是存放在一块区域，未初始化的全局变量和初始化的静态变量存储在相邻的一块区域
4. **文字常量区**：常量字符串就是存放于此，程序结束后由系统释放
5. **程序代码区**：存放函数体的二进制代码

**用预处理指令#define声明一个常数，用以表示一年中有多少秒**

#define SECONDS\_PER\_YEAR (60\*60\*24\*365)UL

//UL表示这个无符号长整型，如果不加UL，将会使得一个16位的整型机器产生溢出

* 1
* 2

**写一个标准宏MIN，这个宏输入两个参数并返回一个较小的数**

#define MIN(x,y) ((x>y)?(y):(x))

//这里为什么都要加括号？比如x，y是一个表达式的话，不加括号的话计算顺序可能不对了

* 1
* 2

**宏定义的函数与普通函数的区别**

宏与函数的选择实际上是时间与空间的权衡。  
宏产生内联代码，也就是说，在程序中产生语句，如果使用宏20次，则会把20行代码插入程序中，如果使用函数20次，那么 程序中只有一份函数语句的拷贝，因此使用函数可以节约空间。  
但是程序的控制必须转移到函数中并随后返回调用程序，因此这比内联代码花费的时间多，所以使用宏定义的函数节约了时间。