在讲多线程程序设计之前，我们先了解一些和进程线程相关的概念

进程：

在Windows早期，一个进程就是一个.exe程序运行时在内存中的模块，包括内存占用空间，主程序代码，以及一些进程参数等。

但是现代OS简化了进程的模型，一个进程就是一个.exe在内存中所有占用的空间，只是一块内存地址，其中32位程序的内存占用空间就是2\*\*32 = 4,294,967,296，即4G，64位程序理论上应用是2\*\*64，但是OS进行了限制，因为这个数字太多，任何程序永远不可能使用这么大的内存空间。

所以我们只需要明白，64位程序占用内存空间是足够大了。

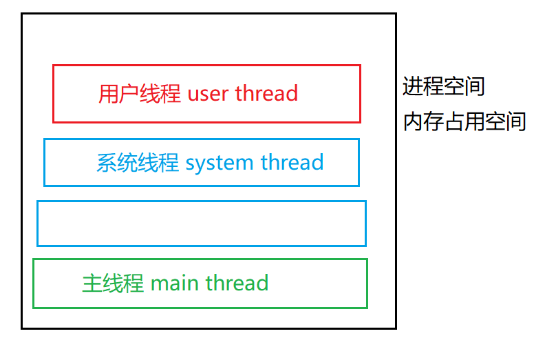
线程：

线程是依附进程存在的，不能脱离进程空间

线程就是程序真正运行的一段代码流，你可以相象：进程就是一间车间，而线程就是车间里真正干活的工人，其中一个工人就代表一个线程，所以线程是可以同时多个存在的。

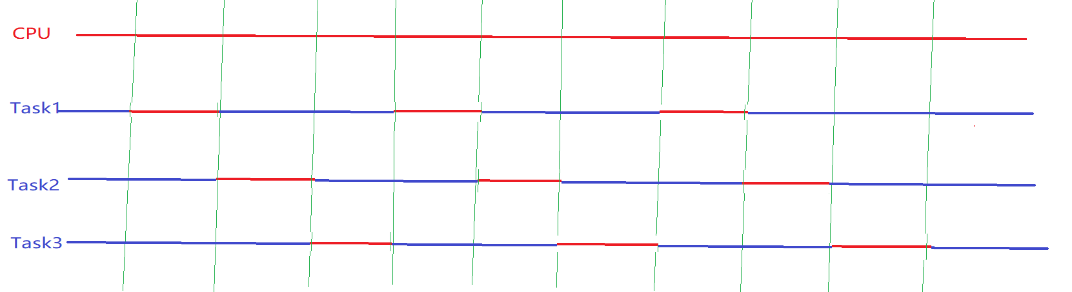
任何一个进程，至少有一条主线程，对我们的程序来说，就是调用main函数（console程序）或winmain函数(GUI程序)的一个代码执行体。

其实对Windows 程序，远不止一个主线程，Windows OS在后台会创建一些线程来辅助做程序背后的事情，但对Windows OS自己创建的线程，我们不需要关心。我们所关心的就是我们自己创建的线程。



为什么需要多线程？

1. 现代CPU本身就是多核的，每一核都是可以独立地，并发地运行。
2. CPU的运行速度远远超过用户的响应速度，如果CPU只与一个用户进行交互响应的话，其实CPU大部分时间都是在等待中。
3. 一般主线程是程序的人机交互界面，需要实时跟人进行互动响应。但有时候会有一些耗时特别长的动作，比如从网络中下载一个大的文件，或者打印一份文档，可能耗时10分钟，那在这个过程中，系统就会卡死，不会响应交互命令。
4. 4，现代OS都是多任务调度系统，可以将CPU的运行快速切换，执行不同的代码，由于这个速度足够快，用户基本上感觉不到切换，会认为是在同时运行



GUI程序的设计原则：

1，主线程是人机交互线程，尽可能处理一些快而小的逻辑，以及和用户界面相关的代码

2，一些费时，繁重的逻辑尽可能放在子线程中处理

3，主线程要管理好子线程的生命周期，主线程结束前，要等待子线程先结束，否则会发现资源等相关物理量没有释放，存在泄漏（包括内存，句柄，资源等）

如何控制子线程提前结束？

这就涉及到线程之间（包括主线程和子线程之间，以及子线程和其它子线程之间）的同步问题。

虽然我们可以不考虑子线程，让主线程直接退出，结束进程，从而释放进程的内存空间（既然整个内存空间都释放了，那么此内存空间中的所有变量，地址等也全都自动释放了）

但是，Windows OS中还有一些对象是内核状态，也不是存放在Windows OS里的，不属于用户的进程空间，这部分是不会自动释放的，除非重启OS。

作为一个资源的，有丰富经验的程序员，必须考虑到所有资源的正确释放！

1. 不考虑到性能的等待，比如sleep
2. 使用线程之间的同步（这个是Senior和Junior的本质区别）

[MFC](https://so.csdn.net/so/search?q=MFC&spm=1001.2101.3001.7020)提供了多种同步对象，最常用的四种：

        1、临界区（CCriticalSection）

        2、事件（CEvent）

        3、互斥量（CMutex）

        4、[信号量](https://so.csdn.net/so/search?q=%E4%BF%A1%E5%8F%B7%E9%87%8F&spm=1001.2101.3001.7020" \t "_blank)（CSemaphore）

前面我们讲过了多线程带来的好处，但它也引来了单线程没有的问题，就是并发访问

之前讲过计算机的CPU只能识别和运行二进制代码，所有的高级语言写的程序都必须通过相关的工具进行转换

高级语言 🡪🡪 汇编代码 🡪🡪 二进制代码

下面只是到汇编代码，如果是二进制代码，将会更多行，也就意味着更多条可执行的语句

Graphical user interface, text, application

Description automatically generated

当某个线程执行到 int c = a + b时，线程发现了切换，这个线程被临时挂起，暂停执行。如果下一个执行线程刚好修改了a或b的值，再当此线程重新执行时，此时的a和b就不再是1和2，当然c的结果也再是3了

而且同时执行的线程越多，OS调度线程的频率越高，线程之间的切换越频繁，就越容易出现上述访问的混乱情况

所以对于线程程序设计，一定要从设计之初就考虑同步问题

1. 线程切换顺序是由OS来控制的，但是程序可以通过一些同步的手段让线程按合理的逻辑运行
2. 主线程是进程内第一个执行的线程，也应该是最后一个退出执行的线程
3. 尽量避免多线程操作同一个变量，如果一定要访问，需要考虑同步问题

有关线程同步的文章，网上很多，一定要多看，多练习，多做笔记，强化训练，积累经验

<https://www.jb51.net/article/215751.htm>

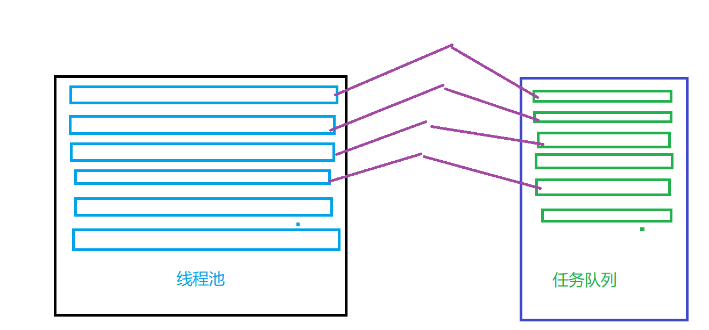
线程池

这是多线程概念的扩展，对于大型程序来说，一定要涉及到多个线程的问题，比如网络访问肯定至少是一个单独的线程，有时候还用多个线程，比如发送是一个线程，接收是另一个线程，对数据的加密也是一个线程，对数据的解密也是一个线程。

对于有人机交互界面的程序来说，交互也是一个线程（对Windows程序来说，一般是主线程，因为主线程是有消息循环的）

另外，程序中还有一些复杂的业务逻辑处理，如果比较费时间，就必须利用线程来处理，而不能阻塞其它线程，特别是主线程

同时，有些线程只是执行一个短时的任务，但是如果我们频繁创建线程，销毁线程，一方面浪费资源，同时也影响性能，所以我们会预先分配好一个线程池，里面一次性创建好所需要的线程，如果线程用完了，也不是销毁，而是将线程挂起，等待后续的重新唤醒。由于所有线程都是在一个线程池中，也方便进行统一管理



线程池是大型Windows程序必须考虑的模型，网上也有大量的文章介绍

https://blog.51cto.com/iteyer/3239144?b=totalstatistic