进程、线程和协程之间的区别和联系

lx青萍之末 2018-10-06 14:56:24 26418 收藏 109

展开

文章目录

一、进程

二、线程

三、进程和线程的区别与联系

四、一个形象的例子解释进程和线程的区别

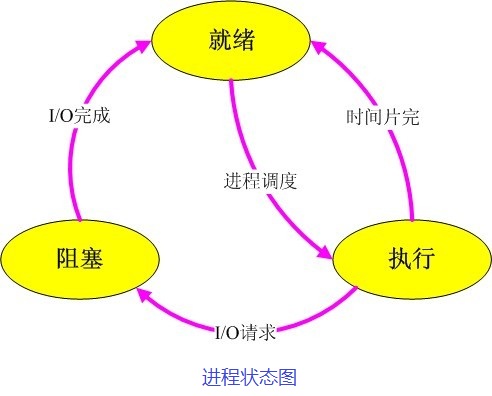
五、进程/线程之间的亲缘性

六、协程

## 一、进程

  进程，直观点说，保存在硬盘上的程序运行以后，会在内存空间里形成一个独立的内存体，这个内存体有自己独立的地址空间，有自己的堆，上级挂靠单位是操作系统。操作系统会以进程为单位，分配系统资源（CPU时间片、内存等资源），进程是资源分配的最小单位。



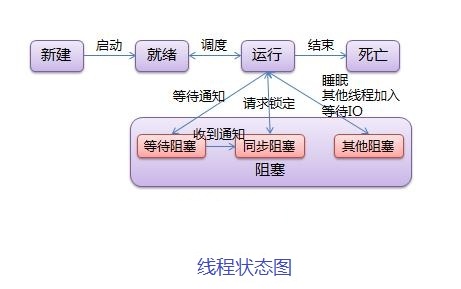


【进程间通信（IPC）】：

管道(Pipe)、命名管道(FIFO)、消息队列(Message Queue) 、信号量(Semaphore) 、共享内存（Shared Memory）；套接字（Socket）。

## 二、线程

  线程，有时被称为轻量级进程(Lightweight Process，LWP），是操作系统调度（CPU调度）执行的最小单位。



## 三、进程和线程的区别与联系

【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，但可以访问隶属于进程的资源。进程所维护的是程序所包含的资源（静态资源）， 如：地址空间，打开的文件句柄集，文件系统状态，信号处理handler等；线程所维护的运行相关的资源（动态资源），如：运行栈，调度相关的控制信息，待处理的信号集等；

系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。但是进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个进程死掉就等于所有的线程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。

【联系】：

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；

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；

处理机分给线程，即真正在处理机上运行的是线程；

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办法实现同步。

## 四、一个形象的例子解释进程和线程的区别



  这副图是一个双向多车道的道路图，假如我们把整条道路看成是一个“进程”的话，那么图中由白色虚线分隔开来的各个车道就是进程中的各个“线程”了。

这些线程(车道)共享了进程(道路)的公共资源(土地资源)。

这些线程(车道)必须依赖于进程(道路)，也就是说，线程不能脱离于进程而存在(就像离开了道路，车道也就没有意义了)。

这些线程(车道)之间可以并发执行(各个车道你走你的，我走我的)，也可以互相同步(某些车道在交通灯亮时禁止继续前行或转弯，必须等待其它车道的车辆通行完毕)。

这些线程(车道)之间依靠代码逻辑(交通灯)来控制运行，一旦代码逻辑控制有误(死锁，多个线程同时竞争唯一资源)，那么线程将陷入混乱，无序之中。

这些线程(车道)之间谁先运行是未知的，只有在线程刚好被分配到CPU时间片(交通灯变化)的那一刻才能知道。

## 五、进程/线程之间的亲缘性

  亲缘性的意思是进程/线程只在某个cpu上运行（多核系统），比如：

BOOL WINAPI SetProcessAffinityMask(

\_In\_ HANDLE hProcess,

\_In\_ DWORD\_PTR dwProcessAffinityMask

);

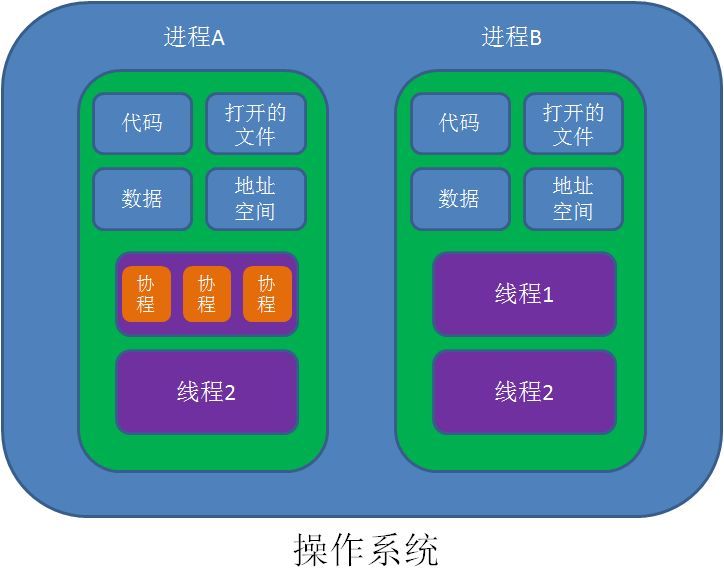
/\*

dwProcessAffinityMask 如果是 0 , 代表当前进程只在cpu0 上工作;

如果是 0x03 , 转为2进制是 00000011 . 代表只在 cpu0 或 cpu1上工作;

  使用CPU亲缘性的好处：设置CPU亲缘性是为了防止进程/线程在CPU的核上频繁切换，从而避免因切换带来的CPU的L1/L2 cache失效，cache失效会降低程序的性能。

## 六、协程



  协程，是一种比线程更加轻量级的存在，协程不是被操作系统内核所管理，而完全是由程序所控制（也就是在用户态执行）。这样带来的好处就是性能得到了很大的提升，不会像线程切换那样消耗资源。

  子程序，或者称为函数，在所有语言中都是层级调用，比如A调用B，B在执行过程中又调用了C，C执行完毕返回，B执行完毕返回，最后是A执行完毕。所以子程序调用是通过栈实现的，一个线程就是执行一个子程序。子程序调用总是一个入口，一次返回，调用顺序是明确的。而协程的调用和子程序不同。

  协程在子程序内部是可中断的，然后转而执行别的子程序，在适当的时候再返回来接着执行。

def A():

print '1'

print '2'

print '3'

def B():

print 'x'

print 'y'

print 'z'

  假设由协程执行，在执行A的过程中，可以随时中断，去执行B，B也可能在执行过程中中断再去执行A，结果可能是：1 2 x y 3 z。

  协程的特点在于是一个线程执行，那和多线程比，协程有何优势？

极高的执行效率：因为子程序切换不是线程切换，而是由程序自身控制，因此，没有线程切换的开销，和多线程比，线程数量越多，协程的性能优势就越明显；

不需要多线程的锁机制：因为只有一个线程，也不存在同时写变量冲突，在协程中控制共享资源不加锁，只需要判断状态就好了，所以执行效率比多线程高很多。

参考：https://blog.csdn.net/tennysonsky/article/details/46046317

https://www.liaoxuefeng.com/wiki/001374738125095c955c1e6d8bb493182103fac9270762a000/0013868328689835ecd883d910145dfa8227b539725e5ed000

https://www.cnblogs.com/work115/p/5620272.html

https://blog.csdn.net/liu251890347/article/details/38509943

https://www.cnblogs.com/fah936861121/articles/8043187.html

http://blog.chinaunix.net/uid-25601623-id-5095687.html

————————————————

版权声明：本文为CSDN博主「lx青萍之末」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/daaikuaichuan/java/article/details/82951084