浅浅念

生如夏花之绚烂,死若秋叶之静美

进程与线程

进程

我们都知道计算机的核心是CPU，它承担了所有的计算任务，而操作系统是计算机的管理者，它负责任务的调度，资源的分配和管理，

统领整个计算机硬件；应用程序是具有某种功能的程序，程序是运行于操作系统之上的。

进程是一个具有一定独立功能的程序在一个数据集上的一次动态执行的过程，是操作系统进行资源分配和调度的一个独立单位，是应用

程序运行的载体。进程是一种抽象的概念，从来没有统一的标准定义。进程一般由程序，数据集合和进程控制块三部分组成。程序用于描述

进程要完成的功能，是控制进程执行的指令集；数据集合是程序在执行时所需要的数据和工作区；程序控制块包含进程的描述信息和控制信息

是进程存在的唯一标志

进程具有的特征：

动态性：进程是程序的一次执行过程，是临时的，有生命期的，是动态产生，动态消亡的；

并发性：任何进程都可以同其他进行一起并发执行；

独立性：进程是系统进行资源分配和调度的一个独立单位；

结构性：进程由程序，数据和进程控制块三部分组成

线程

设计模式(6)

随笔档案

2019年10月(3)
2019年7月(1)
2018年9月(5)
2018年6月(2)
2018年4月(2)
2018年2月(4)
2017年7月(2)
2017年6月(5)
2017年5月(4)
2017年3月(22)
2016年11月(8)
2016年10月(10)

最新评论

1. Re:Git diff 常见用法

git diff --name-only
只列出修改的文件路径

--Dailoge

2. Re:Git push 常见用法

用webRTC做应用可以参考下永久免费的 github.com/starRTC 私有部署挺简单的。

--github.com/starRTC

3. Re:Git diff 常见用法

用webRTC做应用可以参考下永久免费的 github.com/starRTC 私有部署挺简单的。

--github.com/starRTC

4. Re:GitHub 简单用法

感谢哈哈哈 通过你的教程第一次push了自己代码到github

--红烧肉热馒头

5. Re:Git diff 常见用法

加油

--Jaye118

阅读排行榜

1. Git push 常见用法(226815)
2. Git diff 常见用法(84670)
3. Git commit 常见用法(80059)
4. Git branch && Git checkout常见用法(63622)
5. Git clone 常见用法(47194)

在早期的操作系统中并没有线程的概念，进程是拥有资源和独立运行的最小单位，也是程序执行的最小单位。任务调度采用的是时间片

轮转的抢占式调度方式，而进程是任务调度的最小单位，每个进程有各自独立的一块内存，使得各个进程之间内存地址相互隔离。

后来，随着计算机的发展，对CPU的要求越来越高，进程之间的切换开销较大，已经无法满足越来越复杂的程序的要求了。于是就发明

了线程，线程是程序执行中一个单一的顺序控制流程，是程序执行流的最小单元，是处理器调度和分派的基本单位。一个进程可以有一个或

多个线程，各个线程之间共享程序的内存空间(也就是所在进程的内存空间)。一个标准的线程由线程ID，当前指令指针PC，寄存器和堆栈组

成。而进程由内存空间(代码，数据，进程空间，打开的文件)和一个或多个线程组成。

进程与线程的区别

1. 线程是程序执行的最小单位，而进程是操作系统分配资源的最小单位;

2. 一个进程由一个或多个线程组成，线程是一个进程中代码的不同执行路线

3. 进程之间相互独立，但同一进程下的各个线程之间共享程序的内存空间(包括代码段，数据集，堆等)及一些进程级的资源(如打开文件和信

号等)，某进程内的线程在其他进程不可见;

4. 调度和切换：线程上下文切换比进程上下文切换要快得多

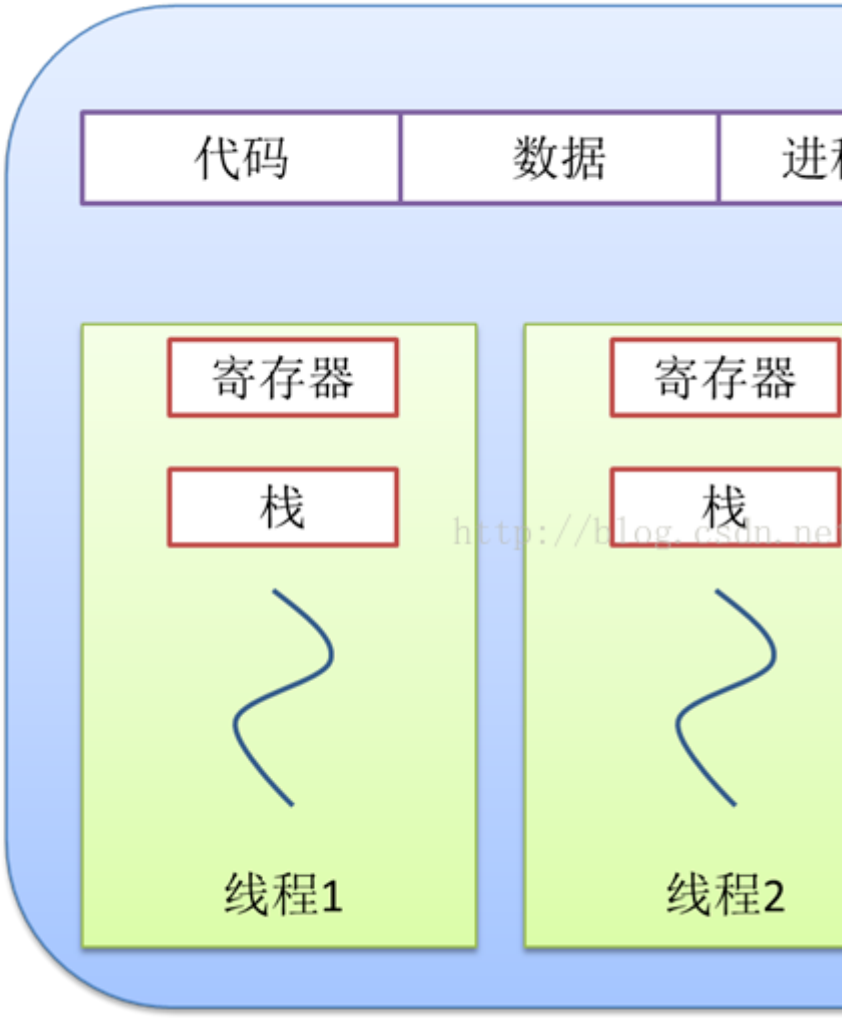
线程和进程关系示意图

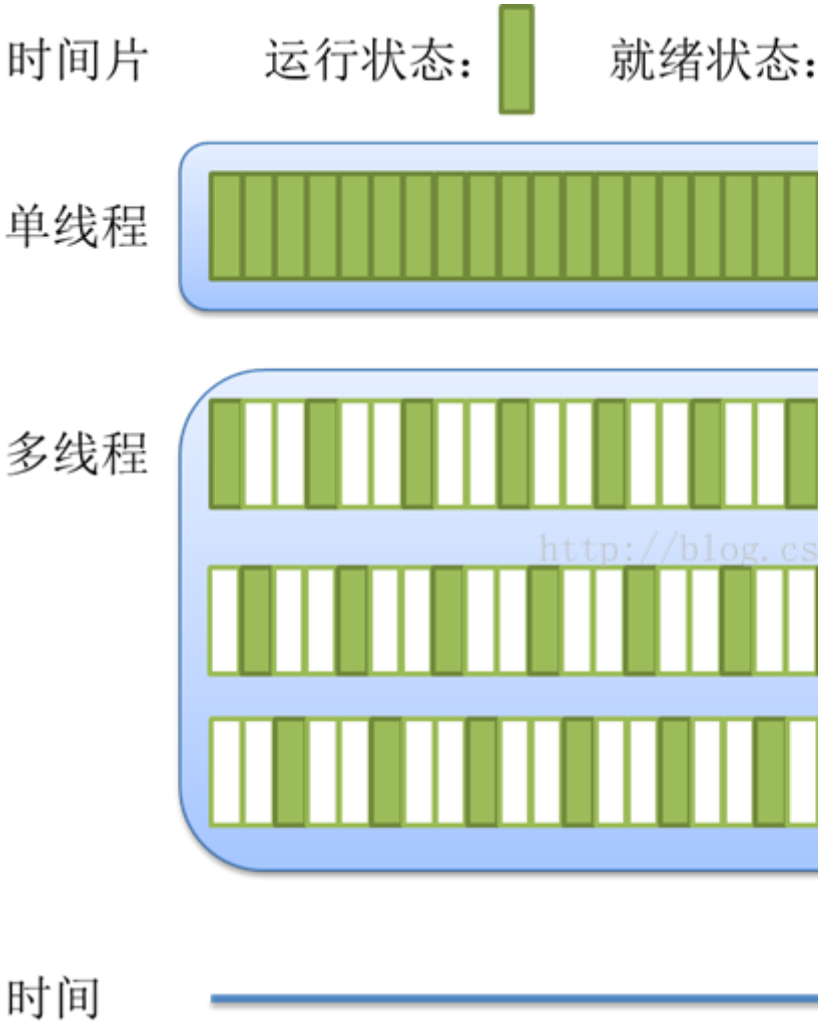
评论排行榜

1. Git diff 常见用法(3)
2. MVC模式和MVP模式的区别(3)
3. 类成员函数可以为回调函数吗(2)
4. Vim 常用简单命令(2)
5. GitHub 简单用法(1)

推荐排行榜

1. Git diff 常见用法(8)
2. Git branch && Git checkout常见用法(4)
3. Git push 常见用法(4)
4. Git commit 常见用法(1)
5. C++三大特性 封装 继承 多态(1)





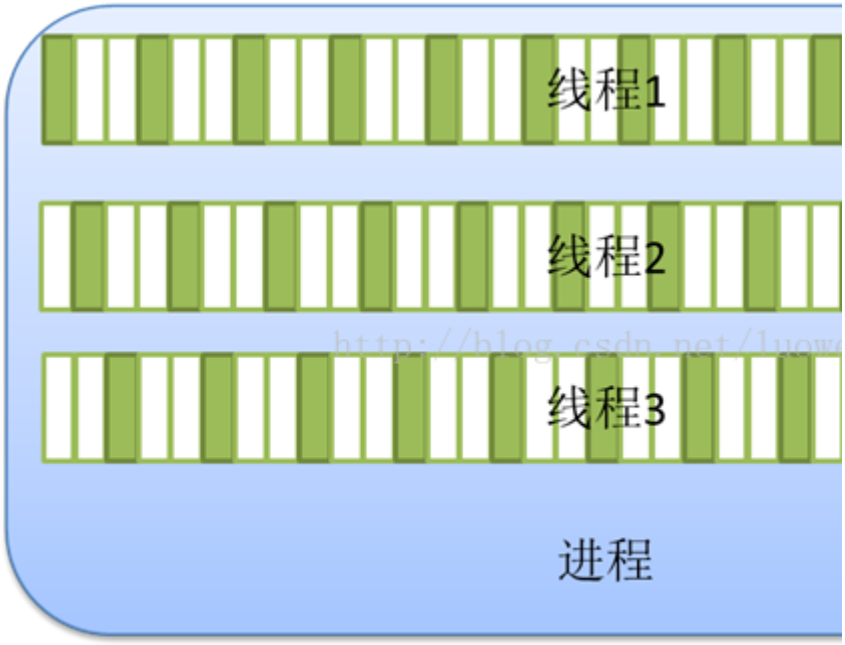
总之，线程和进程都是一种抽象的概念，线程是一种比进程还小的抽象，线程和进程都可用于实现并发。

在早期的操作系统中并没有线程的概念，进程是能拥有资源和独立运行的最小单位，也是程序执行的最小单位，它相当于

一个进程里只有一个线程，进程本身就是线程。所以线程有时被称为轻量级进程

后来，随着计算机的发展，对多个任务之间上下文切换的效率要求越来越高，就抽象出一个更小的概念-线程，一般一个进程会有多个

(也可以是一个)线程。



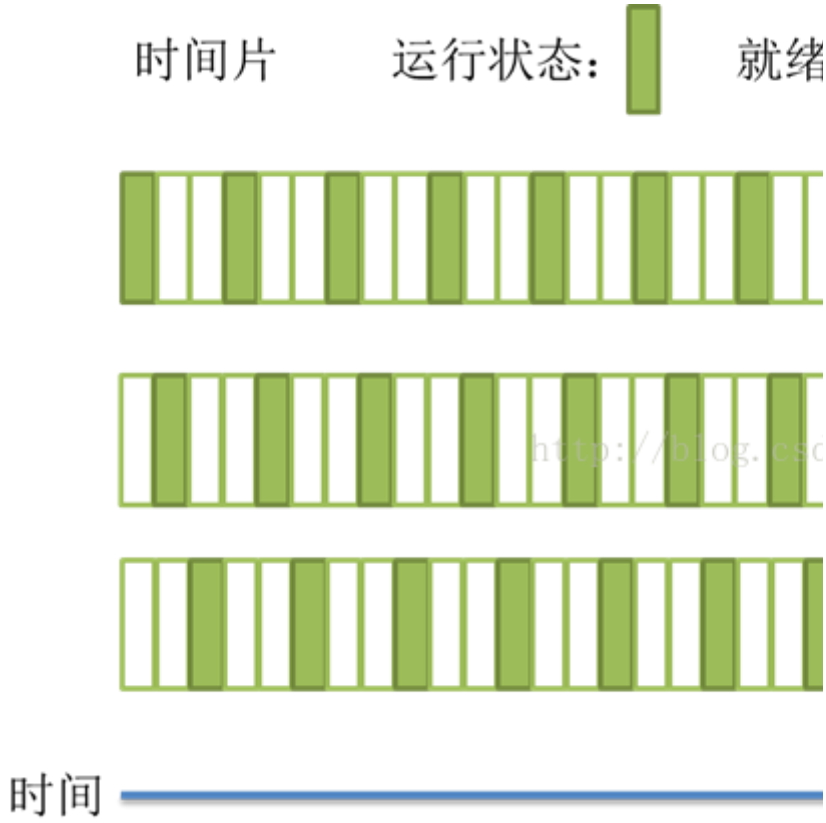
任务调度

大部分操作系统的任务调度是采用时间片轮转的抢占式调度方式，也就是说一个任务执行一小段时间后强制暂停去执行下一个任务，每个

任务轮流执行。任务执行的一小段时间叫做时间片，任务正在执行时的状态叫运行状态，任务执行一段时间后强制暂停去执行下一个任务，被

暂停的任务就处于就绪状态，等待下一个属于它的时间片的到来。这样每个任务都能得到执行，由于CPU的执行效率非常高，时间片

非常短，在各个任务之间快速地切换，给人的感觉就是多个任务在"同时进行"，这也就是我们所说的并发



为何不使用多进程而是使用多线程?

线程廉价，线程启动比较快，退出比较快，对系统资源的冲击也比较小。而且线程彼此分享了大部分核心对象(File Handle)的拥有权

如果使用多重进程，但是不可预期，且测试困难

学习无他法，唯有持之以恒

分类: C++学习

« 上一篇: 设计模式之观察者模式
» 下一篇: 配置Beyond Compare作为比较和合并工具

posted on 2017-07-10 14:04 浅浅念 阅读(9461) 评论(0) 编辑 收藏