分类：理解计算机

上一篇：熵的社会学意义

下一篇：字符串匹配的KMP算法

进程与线程的一个简单解释

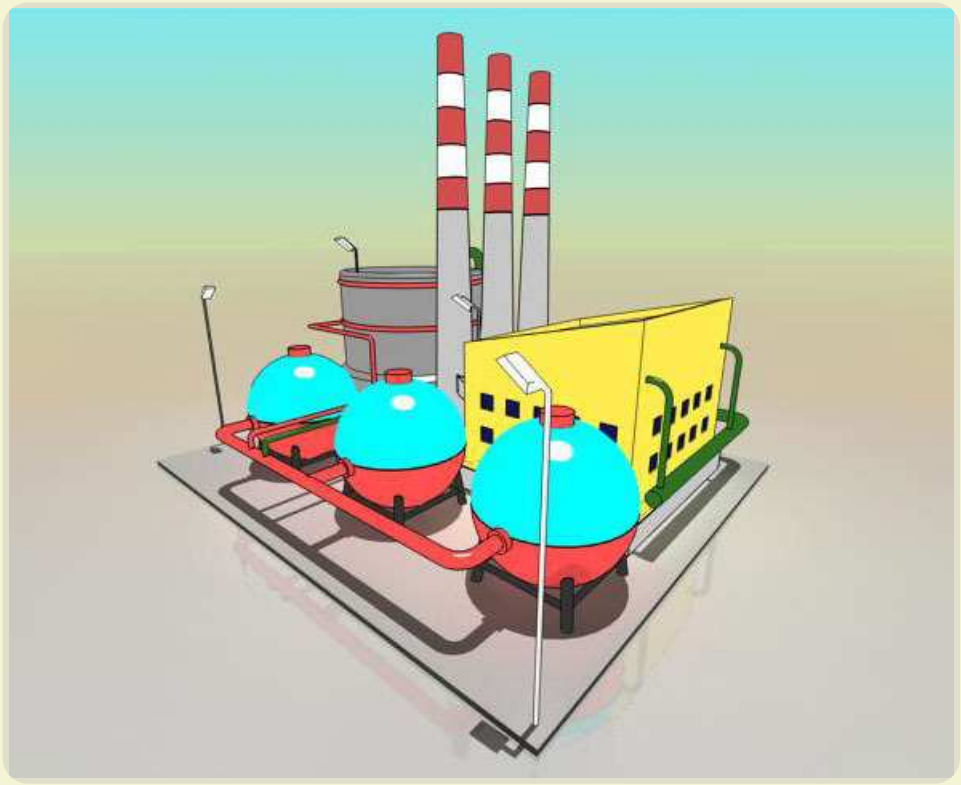
作者：阮一峰

日期：2013年4月24日

进程（process）和线程（thread）是操作系统的基本概念，但是它们比较抽象，不容易掌握。

最近，我读到一篇材料，发现有一个很好的类比，可以把它们解释地清晰易懂。

1.

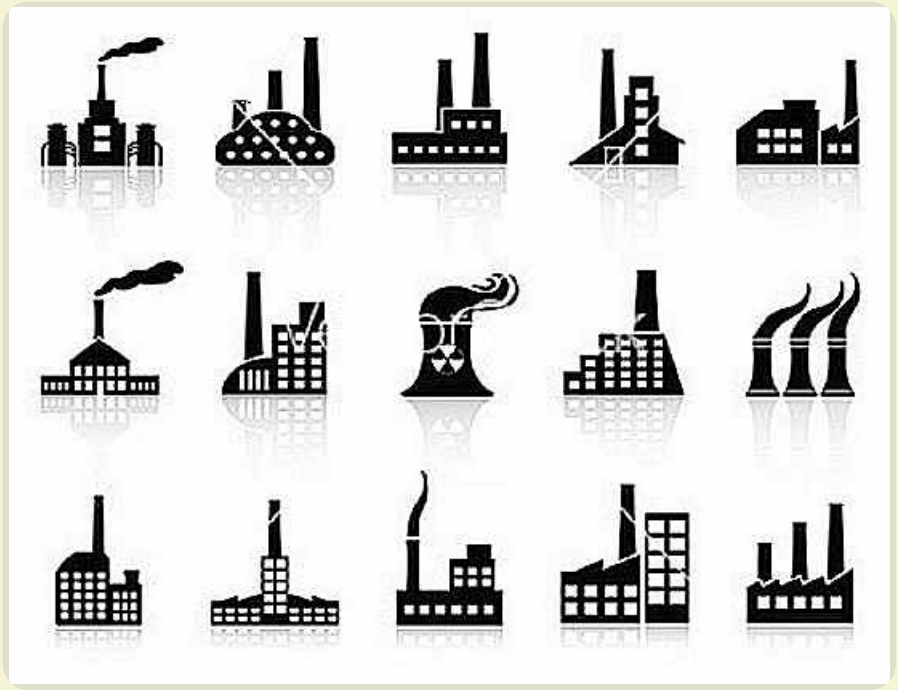


计算机的核心是CPU，它承担了所有的计算任务。它就像一座工厂，时刻在运行。

2.

假定工厂的电力有限, 一次只能供给一个车间使用。也就是说, 一个车间开工的时候, 其他车间都必须停工。背后的含义就是, 单个CPU一次只能运行一个任务。

3.



进程就好比工厂的车间, 它代表CPU所能处理的单个任务。任一时刻, CPU总是运行一个进程, 其他进程处于非运行状态。

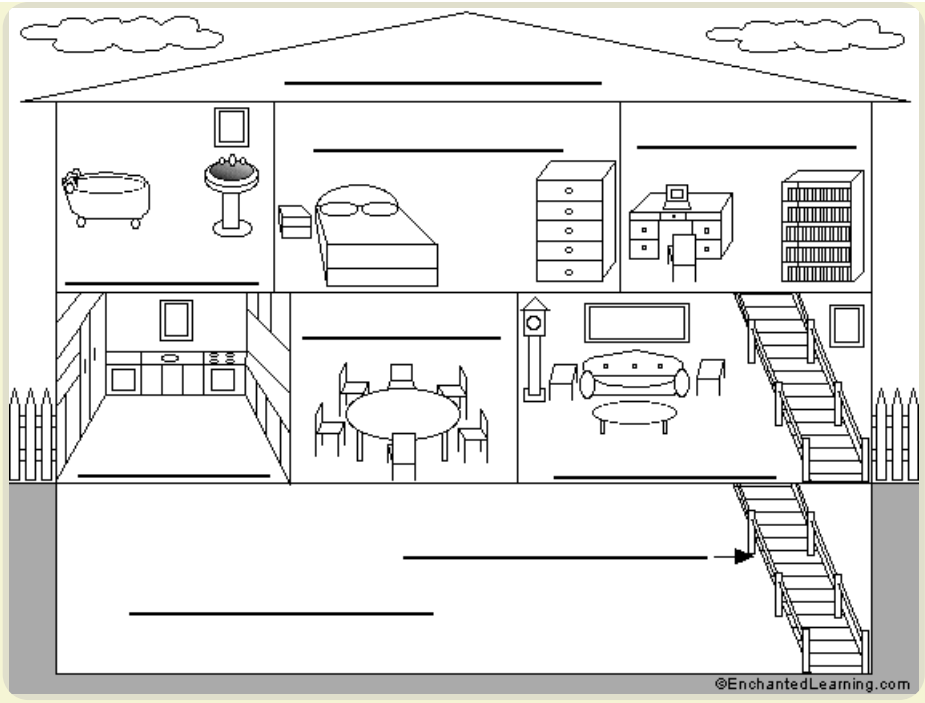
4.

一个车间里, 可以有很多工人。他们协同完成一个任务。

5.

线程就好比车间里的工人。一个进程可以包括多个线程。

6.



车间的空间是工人们共享的, 比如许多房间是每个工人都可以进出的。这象征一个进程的内存空间是共享的, 每个线程都可以使用这些共享内存。

7.

可是, 每间房间的大小不同, 有些房间最多只能容纳一个人, 比如厕所。里面有人的时候, 其他人就不能进去了。这代表一个线程使用某些共享内存时, 其他线程必须等它结束, 才能使用这一块内存。

8.

一个防止他人进入的简单方法, 就是门口加一把锁。先到的人锁上门, 后到的人看到上锁, 就在门口排队, 等锁打开再进去。这就叫"互斥锁"（Mutual exclusion, 缩写 Mutex）, 防止多个线程同时读写某一块内存区域。

9.

还有些房间, 可以同时容纳n个人, 比如厨房。也就是说, 如果人数大于n, 多出来的人只能在外面等着。这好比某些内存区域, 只能供给固定数目的线程使用。

10.

这时的解决方法，就是在门口挂n把钥匙。进去的人就取一把钥匙，出来时再把钥匙挂回原处。后到的人发现钥匙架空了，就知道必须在门口排队等着了。这种做法叫做"信号量"（Semaphore），用来保证多个线程不会互相冲突。

不难看出，mutex是semaphore的一种特殊情况（n=1时）。也就是说，完全可以用后者替代前者。但是，因为mutex较为简单，且效率高，所以在必须保证资源独占的情况下，还是采用这种设计。

11.

操作系统的设计，因此可以归结为三点：

（1）以多进程形式，允许多个任务同时运行；

（2）以多线程形式，允许单个任务分成不同的部分运行；

（3）提供协调机制，一方面防止进程之间和线程之间产生冲突，另一方面允许进程之间和线程之间共享资源。

（完）

- 发表日期: 2013年4月24日

相关文章

- **2021.01.27:** 异或运算 XOR 教程

  > 大家比较熟悉的逻辑运算, 主要是"与运算"(AND)和"或运算"(OR), 还有一种"异或运算"(XOR), 也非常重要。

- **2019.11.17:** 容错, 高可用和灾备

  > 标题里面的三个术语, 很容易混淆, 专业人员有时也会用错。

- **2019.11.03:** 关于计算机科学的50个误解

  > 计算机科学(Computer Science, 简称 CS)是大学的热门专业。但是, 社会上对这个专业有很多误解, 甚至本专业的学生也有误解。

- **2019.10.29:** 你所不知道的 AI 进展

  > 人工智能现在是常见词汇, 大多数人可能觉得, 它是学术话题, 跟普通人关系不大。

留言(121条)

tingliang 说:

写得太好了, 显浅易懂。

2013年4月24日 11:04 | # | 引用

dyb 说:

还不错

2013年4月24日 11:13 | # | 引用

sun 说:

简洁, 浅显易懂, 到时往往越是这样越难以触及本质。当然对于深入理解还是很有帮助的。

2013年4月24日 11:16 | # | 引用

dee 说:

写得太好了 好形象

2013年4月24日 11:23 | # | 引用

hope 说:

配图好亮~

2013年4月24日 11:31 | # | 引用

hejianchao 说:

浅显易懂, 境界真高。

2013年4月24日 11:37 | # | 引用

reverland 说：

喜欢配图～

2013年4月24日 11:49 | # | 引用

Tim Shen 说：

现在说的线程都是内核线程, 是服从系统调度, 独占CPU的。

2013年4月24日 11:50 | # | 引用

herodot 说：

阮兄,
我是你的博客(包括你的英文BLOG)的忠实读者。这个回复纯属技术探讨。

你这篇关于进程和线程的文章, 图文并茂, 很通俗, 但不易懂。坦白说, 在我看来很多比喻很不恰当, 比如:
1. 车间/电力/人三者, 来比喻进程/CPU/线程三者, 其实既没有体现出进程作为程序的一次执行(有独立的内存空间)这一特点, 也没有体现出线程作为CPU调度单位这一特点。人和电力的关系是什么呢?所以“车间的空间是工人们共享的, 比如许多房间是每个工人都可以进出的”, 这句话就很难理解。
2. 用厕所容量, 来比喻锁, 也不合适。因为线程间同步是为了防止竞争(就是说因同时修改, 而导致的数据不一致)。本质上讲, 进程的内存空间是天然独立的, 线程的内存空间是天然共享的。正因为如此, 进程通信/线程同步才是系统编程的很大一块内容。

进程和线程简单而基本靠谱的定义如下:
1. 进程:程序的一次执行
2. 线程:CPU的基本调度单位
这两个概念虽然过于简单, 但是完全可以为理解OS/线程/进程打下坚实的基础, 我认为关于进程/线程的探讨, 无论采用何种方式, 都必须以这两句话为落脚点, 才算靠谱。

你写的大多数文章相当出色。在以通俗的方式解释复杂的概念这方面, 尤其出色。但是, 我个人的经验, 这种方式比较适合用来解释一个复杂概念的一个方面, 或者从某个角度的理解。想面面俱到的说明白一个复杂的系统, 繁琐枯燥的概念, 以及这些概念间的推演几乎不可避免。坦白说, 在实际项目中大量用到线程/进程之前, 我看过很多遍相关的概念, 也写过小的示例程序, 但真到了用的时候, 还是发现之前没搞明白。

我对进程/线程的理解也很粗浅。个人见解, 仅供参考。

2013年4月24日 11:57 | # | 引用

jas 说：

其实线程进程的东西比较好理解, 希望下一期能用形象的方式讲讲协程　纤程

2013年4月24日 12:08 | # | 引用

AriesDevil 说：

写的真好

2013年4月24日 12:43 | # | 引用

cangzhang 说：

期待下一期!

2013年4月24日 12:51 | # | 引用

wolfwd 说：

wallred 说：

没有讲解多进程, 以及CPU对多线程的调度, 不过这些比喻能让人有一些最起码的认识

2013年4月24日 12:52 | # | 引用

尹良灿 说：

刚好这个学期学的操作系统, 表示很有帮助

2013年4月24日 12:56 | # | 引用

大脸猫 说：

我正学到多线程呢?看完以后表示对多线程的理解包括防锁死的机制理解清晰不少。

2013年4月24日 13:06 | # | 引用

chemandy 说：

赞!果真浅显易懂!

2013年4月24日 13:10 | # | 引用

detailyang 说：

多次看到博主的写的文章通俗易懂, 相当优雅啊

2013年4月24日 13:46 | # | 引用

GameXG 说：

>>假定工厂的电力有限, 一次只能供给一个车间使用。也就是说, 一个车间开工的时候, 其他车间都必须停工。背后的含义就是, 单个CPU一次只能运行一个任务。

>>进程就好比工厂的车间, 它代表CPU所能处理的单个任务。任一时刻, CPU总是运行一个进程, 其他进程处于非运行状态。

>>一个车间里, 可以有很多工人。他们协同完成一个任务。

>>线程就好比车间里的工人。一个进程可以包括多个线程。

文章的意思是同一个进程的多个线程是同时执行的, cpu的调度单位是进程。但实际是错误的, cpu的调度单位是线程, 一次只能执行一个线程(当然多核的有几核就能同时执行几个线程), 而不是一次执行一个进程!

2013年4月24日 14:21 | # | 引用

viho_he 说：

抛开各种技术细节, 从应用程序角度讲:

1、在单核计算机里, 有一个资源是无法被多个程序并行使用的:cpu。

没有操作系统的情况下, 一个程序一直独占着全都cpu。

如果要有两个任务来共享同一个CPU, 程序员就需要仔细地为程序安排好运行计划--某时刻cpu和由程序A来独享, 下一时刻cpu由程序B来独享

而这种安排计划后来成为OS的核心组件, 被单独名命为“scheduler”, 即“调度器”, 它关心的只是怎样把单个cpu的运行拆分成一段一段的“运行片”, 轮流分给不同的程序去使用, 而在宏观上, 因为分配切换的速度极快, 就制造出多程序并行在一个cpu上的假象。

2、在单核计算机里, 有一个资源可以被多个程序共用, 然而会引出麻烦:内存。

在一个只有调度器, 没有内存管理组件的操作系统上, 程序员需要手工为每个程序安排运行的空间 -- 程序A使用物理地址0x00-0xff,程序B使用物理地址0x100-0x1ff, 等等。

然而这样做有个很大的问题:每个程序都要协调商量好怎样使用同一个内存上的不同空间, 软件系统和硬件系统千差万别, 使这种定制的方案没有可行性。
为了解决这个麻烦, 计算机系统引入了“虚拟地址”的概念, 从三方面入手来做:

2.1、硬件上, CPU增加了一个专门的模块叫MMU, 负责转换虚拟地址和物理地址。
2.2、操作系统上, 操作系统增加了另一个核心组件:memory management, 即内存管理模块, 它管理物理内存、虚拟内存相关的一系列事务。
2.3、应用程序上, 发明了一个叫做【进程】的模型, (注意)每个进程都用【完全一样的】虚拟地址空间, 然而经由操作系统和硬件MMU协作, 映射到不同的物理地址空间上。不同的【进程】, 都有各自独立的物理内存空间, 不用一些特殊手段, 是无法访问别的进程的物理内存的。

3、现在, 不同的应用程序, 可以不关心底层的物理内存分配, 也不关心CPU的协调共享了。然而还有一个问题存在:有一些程序, 想要共享CPU, 【并且还要共享同样的物理内存】, 这时候, 一个叫【线程】的模型就出现了, 它们被包裹在进程里面, 在调度器的管理下共享CPu, 拥有同样的虚拟地址空间, 同时也共享同一个物理地址空间, 然而, 它们无法越过包裹自己的进程, 去访问别一个进程的物理地址空间。

4、进程之间怎样共享同一个物理地址空间呢?不同的系统方法各异, 符合posix规范的操作系统都提供了一个接口, 叫mmap, 可以把一个物理地址空间映射到不同的进程中, 由不同的进程来共享。

5、PS:在有的操作系统里, 进程不是调度单位(即不能被调度器使用), 线程是最基本的调度单位, 调度器只调度线程, 不调度进程, 比如VxWorks

2013年4月24日 14:53 | # | 引用

selfcontrol 说:

这个文章是转贴的。我认为楼主转贴的有些仓促了, 浅显易懂但是根本不准确。很多比喻都错了。

2013年4月24日 15:13 | # | 引用

selfcontrol 说:

> **引用herodot的发言:**
>
> 你写的大多数文章相当出色。在以通俗的方式解释复杂的概念这方面, 尤其出色。但是, 我个人的经验, 这种方式比较适合用来解释一个复杂概念的一个方面, 或者从某个角度的理解。想面面俱到的说明白一个复杂的系统, 繁琐枯燥的概念, 以及这些概念间的推演几乎不可避免。坦白说, 在实际项目中大量用到线程/进程之前, 我看过很多遍相关的概念, 也写过小的示例程序, 但真到了用的时候, 还是发现之前没搞明白。

握手, 比喻确实浅显易懂, 但是很容易导致谬误。

另外, 这篇文章是作者转贴的, 不是他原创的。

2013年4月24日 15:15 | # | 引用

Tux 说:

> **引用GameXG的发言:**
>
> 文章的意思是同一个进程的多个线程是同时执行的, cpu的调度单位是进程。但实际是错误的, cpu的调度单位是线程, 一次只能执行一个线程(当然多核的有几核就能同时执行几个线程), 而不是一次执行一个进程!

对, 这样解释其实更容易误解。进程和线程的区别其实也没那么明显, 像linux里面就只有一个task, 当

然也可以根据不同task间共享数据的多少来区分是否线程, 不过似乎没什么必要。

2013年4月24日 16:34 | # | 引用

阮一峰 说：

> **引用selfcontrol的发言：**
>
> 这个文章是转贴的。

你是指从英文翻译的?

2013年4月24日 16:41 | # | 引用

阮一峰 说：

> **引用herodot的发言：**
>
> 你这篇关于进程和线程的文章, 图文并茂, 很通俗, 但不易懂。坦白说, 在我看来很多比喻很不恰当
>
> 进程和线程简单而基本靠谱的定义如下:
> 1. 进程:程序的一次执行
> 2. 线程:CPU的基本调度单位

我完全同意你对进程和线程的定义。

这篇文章的比喻, 确实不是完全准确。但是, 进程和线程要找到现实生活中准确的对应, 恐怕很困难吧。

我的基本想法是:

(1)进程是一个容器。

(2)线程是容器中的工作单位。

车间—工人的比喻就是这样来的。

2013年4月24日 16:47 | # | 引用

jeffhyj 说：

我觉得解释的很好也很正确啊, 只是没有说明多进程这种情况。

2013年4月24日 16:49 | # | 引用

xgl 说：

配图挺逗, 哈哈

2013年4月24日 17:26 | # | 引用

dota 说：

不是很容易让人理解。留言里面有高手

2013年4月24日 20:00 | # | 引用

herodot 说：

> **引用阮一峰的发言：**

> 我完全同意你对进程和线程的定义。
>
> 这篇文章的比喻, 确实不是完全准确。但是, 进程和线程要找到现实生活中准确的对应, 恐怕很困难吧。
>
> 我的基本想法是:
>
> (1)进程是一个容器。
>
> (2)线程是容器中的工作单位。
>
> 车间—工人的比喻就是这样来的。

确实, 对于一个复杂点的系统来说, 在现实中找到完全对位的例子很难。这也是我前面回复中提到的, 类比这种表达方式, 贵在准而不在全。用一个生活中的例子, 捕获到复杂系统一个侧面, 就足够了。想完全对位, 取舍之后, 往往会很牵强, 失去通俗易懂的目的。

我大概浏览了一下你所参考的原文(原文很长, 我英文一般, 可能理解的不完整), 发现原文重点讲述了三个方面:
1. 进程与线程的关系
2. 线程间的关系
3. 线程与CPU (核心)的关系
而且只有1和2是用类比(多人合租房子, 房子比做进程, 人比做线程)来讲解的, 这种比喻可以说很恰当, 把进程和线程间的关系, 以及线程间的关系分析的非常清楚。而讲到3的时候, 已经更多的从概念角度分析了, 就是说在作者并没有试图在一个类比中说明这三个概念(我没有在那篇文章中找到电力概念)。

而阮兄的文章, 通过 工厂/车间/人 试图把CPU/进程/线程说清楚, 这也是我的困惑所在。
在我看了, 引入电力这种类比之后, 但是开头这句, 已经把线程/进程混淆了(不过话说回来, 在linux系统里面, 在最底层, 进程线程确实不分的):

> 假定工厂的电力有限, 一次只能供给一个车间使用。也就是说, 一个车间开工的时候, 其他车间都必须停工。背后的含义就是, 单个CPU一次只能运行一个任务。

BTW:

2013年4月24日 21:11 | # | 引用

小涛 说:

> 进程是OS在程序运行时资源分配和调度的一个独立单元, 一个进程可以开启一个或多个线程, 进程抢到cpu时间片之后, 如果该进程开启了多个线程, 再由线程抢占cpu时间片, 谁得到时间片谁执行。

2013年4月24日 21:42 | # | 引用

86er 说:

这个写的太赞了, 通俗易懂!

2013年4月24日 21:46 | # | 引用

selfcontrol 说:

> **引用阮一峰的发言:**
>
> 你是指从英文翻译的?

我的意思就是从翻译来的。我用词错误, 用转贴是不合适的。见谅见谅。

2013年4月25日 01:13 | # | 引用

（void*）说：

这篇文章写的很好，尤其是对于对计算机系统了解不是很多的人，看了这篇文章之后，会对计算机中的多任务（其实也就是多进程），以及一个进程的多线程有了一个初步的了解。但是因为文章的更多的内容是关于CPU调度的，所以其中对内存的东西解释的就不是那么清楚了。比如上面各位提到的内存共享之类的讨论。计算器系统对计算资源的分配（CPU）和对内存的分配是分开进行的，只有这两个方面的知识都介绍了，才能比较清楚的解释这方面的内容。相反如果非要把一个柔和到一个中去，那么就会出现各种不合适的地方。比如我觉的用房子的比喻进程，用房间比喻线程，是合适的，将是吧内存比喻为房子中房间是不合适的。毕竟系统的内存的时候用多种不一样的用法，堆和栈你可以勉强的比喻为房间中的空间，但是共享内存，这个东西就在进程之外了，再用空间比喻就不恰当了。我觉的阮兄可以考虑再写一篇系统如何管理内存的问题。

2013年4月25日 10:38 | # | 引用

冰上游鱼 说：

配图很好玩啊

2013年4月25日 10:43 | # | 引用

viho_he 说：

> **引用小涛的发言：**
>
> 进程是OS在程序运行时资源分配和调度的一个独立单元，一个进程可以开启一个或多个线程，进程抢到cpu时间片之后，如果该进程开启了多个线程，再由线程抢占cpu时间片，谁得到时间片谁执行。

就我所知道到的OS(windows / linux / VxWorks / qnx)里，目前没有哪个是这样做的。

虽然目前没有不排除有这系统会这样去实现，然而从模型来说，这是个很糟糕的模型，因为这个模型增加了调度器的复杂度，调度器需要判断当前调度的单位是进程还是线程，并且做出不同的处理，加上优先级、时间片轮转、线程/进程运行状态等机制以后，此模型复杂度会显著增加，而到了SMP（同构并发多处理器）的系统上，此模型基本上已无能够工作的可能性。

即使在qnx这样鼓励用进程而非线程的微内核的OS上，文档仍然说到：Threads are scheduled globally across all processes.即调度的基本单位是线程，不是进程。调度器在调度任务时，只知道线程，不知道进程这个东西，对于进程的管理，是OS别的组件的事情。

2013年4月25日 12:30 | # | 引用

noname 说：

但实际上，从low-level implementation的角度来看，线程进程没什么区别，只是要处理一些资源共享带来的一些问题(比如信号怎么处理)而已。因此真正design和implement过操作系统的人就会觉得，这么解释反而会增加一些误解。

2013年4月26日 23:19 | # | 引用

六月 说：

从调度来看，进程和线程应该是一样的。

2013年4月26日 23:42 | # | 引用

gordon 说：

> **引用viho_he的发言：**

> 就我所知道到的OS(windows / linux / VxWorks / qnx)里, 目前没有哪个是这样做的。

viho_he 能够推荐一些资料吗?受个人条件限制, 能把硬件和操作系统结合起来讲的人很少, 遇到一个明白人不容易, 就厚着脸皮问了。

2013年4月27日 05:21 | # | 引用

artwl 说:

通俗易懂

2013年4月27日 14:42 | # | 引用

V客小站 说:

浅显易懂, 好文!

2013年4月27日 17:35 | # | 引用

bj0629 说:

我个人感觉80年代和90年代初期的书和杂志里一般说multi task,后来变成 multi processing , 在90年代末期 windows 编程里又看到了 multi thread。单个CPU实现所谓的多任务、进程、线程, 只能靠分时中断或事件中断, 排队来实现。有些理念我自己理解是在忽悠和为了广告效应而已。

谢谢楼主的分享

2013年4月28日 09:45 | # | 引用

gordon 说:

为什么要引入多级存储和虚拟存储器?

单级存储器的基本矛盾和解决方法

在一定存储器件条件下, 容量越大, 其寄生参量和走线延迟也越大, 因而会降低其工作主频, 也就是降低了存储速度。

寄生参量是电路的东西, 为什么要考虑这个?

最典型的就是现在最热门的量子计算机, 根据摩尔定律 大概每两年 一块集成电路板上可以容纳的器件数会加倍。这个大家可以感受得到 计算机和其他电子设备的速度容量都在不断翻倍 当一个二级管小到只有几个原子那么大的时候 经典物理就不再适用 必须考虑量子效应了传统计算机10几年后就会碰到量子障碍。

2013年4月29日 18:52 | # | 引用

xubincd 说:

话说, 我认为抽象的概念, 发明出多少种比喻来进行解释也是在做无用功, 抽象就是抽象, 必须经常用去琢磨这种概念, 拿现实世界的来做比喻我觉得对抽象本身来说其实并无帮助。
仅个人意见, 不知道各位怎么看

2013年4月29日 22:34 | # | 引用

viho_he 说:

> **引用gordon的发言:**
>
> viho_he 能够推荐一些资料吗?受个人条件限制, 能把硬件和操作系统结合起来讲的人很少, 遇到一个明白人不容易, 就厚着脸皮问了。

操作系统是一门实践性极强的学问, 而且与CPU的硬件功能结合得非常紧密, 可以说操作系统的核心模块都是关于怎样玩转CPU的硬件功能的, 很多过去原本是操作系统的软件功能, 被做到了CPU硬件里面。个人经验是, 不要怕接触技术细节, 只有在大量具体的技术细节的积累之上, 才能真正理解抽像的概念。我建议你调试linux kernel玩玩, 有条件的话买一个arm的开发板自己弄弄, 没条件可以在PC机上安装模拟器来玩, 比如QEMU。读书的话, 可以啃《Linux内核源代码情景分析》, 不用读全书, 只把你关心的那几章通读就行了。另外, 为了能真正读懂kernel源码, 你需要至少学习一门CPU的知识（即阅读枯燥冗长的CPU手册）作为辅助, 可以从IA入手, 也可以从ARM入手, 以你手头的能用的硬件为准。最起码要了解中断的处理机制、内存的处理机制。《Linux内核源代码情景分析》用的是IA。

2013年5月 2日 15:12 | # | 引用

lee 说：

懂得人一看就懂
不懂的人就.....

2013年5月 2日 18:43 | # | 引用

红色石头 说：

配图不错~~~很有爱...
另外, 我对阮一峰这个名字感觉又印象, 问下你是写过什么技术类书籍的?

2013年5月 9日 10:53 | # | 引用

mdyang 说：

“不难看出, mutex是semaphore的一种特殊情况（n=1时）。也就是说, 完全可以用后者替代前者。”
mutex和二元信号量并不同, http://stackoverflow.com/questions/62814/difference-between-binary-semaphore-and-mutex

2013年5月 9日 12:17 | # | 引用

土木坛子 说：

能把抽象复杂的现象用通俗易懂的方式讲出来, 是多么不容易的一件事情啊。

2013年5月11日 04:20 | # | 引用

李奇 说：

虽然不是很准确, 不过确实比较形象, 对于初级了解一下挺有帮助。
但还未涉及到进程和线程资源分配的问题, 加上这部分会更完整。

2013年5月21日 00:29 | # | 引用

吕小荣 说：

通俗易懂, 很适合新手理解。

2013年5月24日 09:57 | # | 引用

daleydeng 说：

简单的说, 线程就是深度共享的进程, 都是名为struct task_struct 的一家人 by kernerer

2013年6月 3日 23:51 | # | 引用

屌丝男 说：

进程就是我们打开的一个快播播放器,线程就是快播里面的那些功能,如搜索、播放、暂停等。。

2013年7月 5日 22:06 | # | 引用

helloworld 说：

楼主怎么没给翻译完啊?很期待啊

2013年9月21日 09:38 | # | 引用

刀尖红叶 说：

通俗易懂~

2013年11月23日 15:10 | # | 引用

大鱼 说：

看了留言收获很多。

2013年12月15日 23:22 | # | 引用

pubemail 说：

开始略懂进程和线程

2014年3月12日 14:48 | # | 引用

tuzhi 说：

写得真好。还有一种分法是按照是否拥有独立用户空间。

2014年3月17日 19:38 | # | 引用

ifanr 说：

通俗易懂, 适合资深新手, 呆学得足够深入时, 才好提自己的看法……

2014年4月 3日 17:16 | # | 引用

马琳琳 说：

很容易看懂, 图片也放的很好看, 这在为软考备战, 看了这篇文章, 觉得看题看的晕晕的我也不晕了, 谢谢楼主分享这么好的经验, 这么好的文章, 感谢

2014年4月10日 10:27 | # | 引用

coderlu 说：

果然简单易懂, 但是没怎么讲进程喔。

2014年5月 5日 11:38 | # | 引用

barry.zhou 说：

进程和线程的类比总结得非常好, 跟我之前自己总结的一样, 知音啊~

2014年6月 7日 10:41 | # | 引用

Alina1999 说：

确实很形象

2014年7月 2日 11:53 | # | 引用

Joanna 说：

写的特别赞!!!很形象很好懂

2014年9月21日 17:06 | # | 引用

Ssun 说：

醍醐灌顶的感觉, 我很喜欢看博主的文章, 看完发现自己需要学的东西太多了!

2014年9月30日 21:20 | # | 引用

春泥面包 说：

看评论也能学到不少东西

2014年10月10日 09:05 | # | 引用

Hallie 说：

直观形象！

2014年11月19日 17:51 | # | 引用

TK 说：

@viho_he：

寫的太讚了

2014年11月25日 01:28 | # | 引用

问 说：

通俗易懂，写的太好了，期待下一期。这必须要赞一个！！非常好

2014年12月 5日 17:50 | # | 引用

niewj 说：

看你的技术博客，就像听袁腾飞讲历史，难得的技术解惑达人！！！

2015年1月 9日 00:45 | # | 引用

书生Maple 说：

真真是极好的，浅显易懂。以前对这一块特别的混乱，现在理清了这些思路。想到进程、线程，就想到工厂、车间了。

2015年2月 7日 14:25 | # | 引用

一生只做一件事 说：

这种方式比较适合用来解释一个复杂概念的一个方面，或者从某个角度的理解。想面面俱到的说明白一个复杂的系统，繁琐枯燥的概念，以及这些概念间的推演几乎不可避免。正解！
但 阮 先生能解释到这个地步已属难得，尤其是这种比喻方式极好。往往让初学者瞬间理清思路。

2015年3月21日 18:26 | # | 引用

火贼君 说：

大神

2015年3月26日 14:26 | # | 引用

zephoonyu 说：

赞一个

2015年4月 2日 20:58 | # | 引用

yueyanglou 说：

这篇文章写得很好，对于概念比较混淆或者对因为其他文章描写复杂的互斥锁望而却步的人有很大的帮助！博主辛苦了！

2015年4月10日 22:55 | # | 引用

FredWe 说：

> **引用viho_he的发言：**
>
> 抛开各种技术细节, 从应用程序角度讲：
>
> 1、在单核计算机里, 有一个资源是无法被多个程序并行使用的：cpu。
>
> 没有操作系统的情况下, 一个程序一直独占着全都cpu。
>
> 如果要有两个任务来共享同一个CPU, 程序员就需要仔细地为程序安排好运行计划--某时刻cpu和由程序A来独享, 下一时刻cpu由程序B来独享
>
> 而这种安排计划后来成为OS的核心组件, 被单独名命为“scheduler”, 即“调度器”, 它关心的只是怎样把单个cpu的运行拆分成一段一段的“运行片”, 轮流分给不同的程序去使用, 而在宏观上, 因为分配切换的速度极快, 就制造出多程序并行在一个cpu上的假象。
>
> 2、在单核计算机里, 有一个资源可以被多个程序共用, 然而会引出麻烦：内存。
>
> 在一个只有调度器, 没有内存管理组件的操作系统上, 程序员需要手工为每个程序安排运行的空间 -- 程序A使用物理地址0x00-0xff,程序B使用物理地址0x100-0x1ff, 等等。
>
> 然而这样做有个很大的问题：每个程序都要协调商量好怎样使用同一个内存上的不同空间, 软件系统和硬件系统千差万别, 使这种定制的方案没有可行性。
> 为了解决这个麻烦, 计算机系统引入了“虚拟地址”的概念, 从三方面入手来做：
>
> 2.1、硬件上, CPU增加了一个专门的模块叫MMU, 负责转换虚拟地址和物理地址。
> 2.2、操作系统上, 操作系统增加了另一个核心组件：memory management, 即内存管理模块, 它管理物理内存、虚拟内存相关的一系列事务。
> 2.3、应用程序上, 发明了一个叫做【进程】的模型, (注意) 每个进程都用【完全一样的】虚拟地址空间, 然而经由操作系统和硬件MMU协作, 映射到不同的物理地址空间上。不同的【进程】, 都有各自独立的物理内存空间, 不用一些特殊手段, 是无法访问别的进程的物理内存的。
>
> 3、现在, 不同的应用程序, 可以不关心底层的物理内存分配, 也不关心CPU的协调共享了。然而还有一个问题存在：有一些程序, 想要共享CPU, 【并且还要共享同样的物理内存】, 这时候, 一个叫【线程】的模型就出现了, 它们被包裹在进程里面, 在调度器的管理下共享CPu, 拥有同样的虚拟地址空间, 同时也共享同一个物理地址空间, 然而, 它们无法越过包裹自己的进程, 去访问别一个进程的物理地址空间。
>
> 4、进程之间怎样共享同一个物理地址空间呢?不同的系统方法各异, 符合posix规范的操作系统都提供了一个接口, 叫mmap, 可以把一个物理地址空间映射到不同的进程中, 由不同的进程来共享。
>
> 5、PS：在有的操作系统里, 进程不是调度单位(即不能被调度器使用), 线程是最基本的调度单位, 调度器只调度线程, 不调度进程, 比如VxWorks

2015年4月25日 01:56 | # | 引用

FredWe 说：

这个评论写得好, 想知道对于多核处理器又该如何解释

> **引用viho_he的发言：**

抛开各种技术细节, 从应用程序角度讲：

1、在单核计算机里, 有一个资源是无法被多个程序并行使用的：cpu。

没有操作系统的情况下, 一个程序一直独占着全都cpu。

如果要有两个任务来共享同一个CPU, 程序员就需要仔细地为程序安排好运行计划--某时刻cpu和由程序A来独享, 下一时刻cpu由程序B来独享

...

2015年4月25日 01:57 | # | 引用

lukase 说：

浅显易懂, 境界真高。

2015年5月12日 08:58 | # | 引用

anonymous 说：

> **引用herodot的发言：**
>
> 阮兄,
> 我是你的博客(包括你的英文BLOG)的忠实读者。这个回复纯属技术探讨。
>
> 你这篇关于进程和线程的文章, 图文并茂, 很通俗, 但不易懂。坦白说, 在我看来很多比喻很不恰当, 比如:
> 1. 车间/电力/人三者, 来比喻进程/CPU/线程三者, 其实既没有体现出进程作为程序的一次执行(有独立的内存空间)这一特点, 也没有体现出线程作为CPU调度单位这一特点。人和电力的关系是什么呢?所以"车间的空间是工人们共享的, 比如许多房间是每个工人都可以进出的", 这句话就很难理解。
> 2. 用厕所容量, 来比喻锁, 也不合适。因为线程间同步是为了防止竞争(就是说因同时修改, 而导致的数据不一致)。本质上讲, 进程的内存空间是天然独立的, 线程的内存空间是天然共享的。正因为如此, 进程通信/线程同步才是系统编程的很大一块内容。
>
> 进程和线程简单而基本靠谱的定义如下:
> 1. 进程:程序的一次执行
> 2. 线程:CPU的基本调度单位
> 这两个概念虽然过于简单, 但是完全可以为理解OS/线程/进程打下坚实的基础, 我认为关于进程/线程的探讨, 无论采用何种方式, 都必须以这两句话为落脚点, 才算靠谱。
>
> 你写的大多数文章相当出色。在以通俗的方式解释复杂的概念这方面, 尤其出色。但是, 我个人的经验, 这种方式比较适合用来解释一个复杂概念的一个方面, 或者从某个角度的理解。想面面俱到的说明白一个复杂的系统, 繁琐枯燥的概念, 以及这些概念间的推演几乎不可避免。坦白说, 在实际项目中大量用到线程/进程之前, 我看过很多遍相关的概念, 也写过小的示例程序, 但真到了用的时候, 还是发现之前没搞明白。
>
> 我对进程/线程的理解也很粗浅。个人见解, 仅供参考。

这个才是正解 楼主说的其实我真没看懂 但是在各位仁兄说的 我貌似知道了点什么。

2015年6月19日 11:32 | # | 引用

wisdom701021 说：

這篇文章真的寫得太好了, 跟我心中所想, 完全不謀而合
但願版主下次還有類似的講解
(當然如果這篇文章還有簡單易懂以及可執行的程式碼的話那更好)
期待下次再來, 謝謝分享

2015年8月28日 13:28 | # | 引用

lansekuihua 说：

哈哈, 好形象的比喻, 一下子都看懂了, 顺便理解了锁和信号量的含义!

2015年9月 2日 21:07 | # | 引用

--=---- 说:

解释的很清楚

2015年9月 7日 02:59 | # | 引用

Rick 说:

像读小人书拟的

2015年9月 9日 17:04 | # | 引用

mousycoder 说:

@herodot:

线程其实是轻量级进程(Lightweight Process, LWP), 这个才是根本, 进程才是CPU调度基本单位。

阮兄的比喻确实不够恰当, 其实这些概念翻译过来, 都变味了, 不过你的评判大部分是对的, 少部分还是有些问题。

详情可以看下:http://mousycoder.com/the-pragmatic-sa-13/

2015年9月14日 13:27 | # | 引用

小浩 说:

懂了这部分 想看下多线程

2015年10月 9日 19:41 | # | 引用

混的灵 说:

阮兄弟是写给没接触过的人, 至少很快的理解一些基本概念。各位大神是帮我们纠正错误的概念, 进一步帮我们提高, 所以看完这篇博客和评论, 学习效率很高。学习本来就应该相互探讨, 百家争鸣, 拥有批判精神, 这样你们的过程和结果, 让我们这个菜鸟受益无穷

2015年10月14日 15:10 | # | 引用

ss 说:

我是初学者, 对于进程, 线程, 程序三者概念模糊。老师可以解释一下程序吗?谢谢。

2015年11月23日 09:20 | # | 引用

海星 说:

我见过最最最形象的讲解!太棒了好喜欢~

2015年12月16日 09:02 | # | 引用

MummyDing 说:

> **引用mousycoder的发言:**
>
> @herodot:
>
> 线程其实是轻量级进程(Lightweight Process, LWP), 这个才是根本, 进程才是CPU调度基本单位。
>
> 阮兄的比喻确实不够恰当, 其实这些概念翻译过来, 都变味了, 不过你的评判大部分是对的, 少部分还是有些问题。

> 详情可以看下：http://mousycoder.com/the-pragmatic-sa-13/

博文很赞,不过地址现在变了,更新下 http://mousycoder.com/2015/10/14/the-pragmatic-sa-13/

2015年12月29日 22:16 | # | 引用

张家浩 说：

我理解的进程, 就相当于我们公司就是一个程序, 没接到一个项目就开始了一个进程, 我们工程师就相当于cpu, 单核就相当于只有我这一个工程师, 多进程就相当于我们单位同时接了好多个项目, 我把每一个项目拆分成一个个小的项目来依次完成, 所以这些项目的小项目就是线程, 共享了我和我处理这个项目所需的一些材料, 但是项目之间我是不能混淆共享的, 我会把一天时间分给一个项目, 把这天中的一个个小时分给各个小小项目, 不断地在多个小小项目和项目之间切换, 来给我的上司（用户）呈现出项目都在执行的假象, 不知道这样的理解对不对。

2016年1月 4日 23:06 | # | 引用

sandheart 说：

通俗易懂, 不懂计算机的也能理解

2016年2月 4日 16:43 | # | 引用

mary 说：

> **引用sun的发言：**
>
> 简洁, 浅显易懂, 到时往往越是这样越难以触及本质。当然对于深入理解还是很有帮助的。

你如果觉得难以触及本质大可以自己去看“本质”的书籍, 不要一边在这里看了文章入了门, 一边喷别人写的“不触及本质”。请问您看文章给阮一峰付费了么。吃相难看, 懂?

2016年3月 6日 23:57 | # | 引用

public 说：

写的真好

2016年6月 3日 13:52 | # | 引用

test 说：

只有我觉得完全没用吗?讲给小学生听得??

2016年6月11日 17:22 | # | 引用

jain 说：

搞这么复杂干嘛, 写段代码不就浅显易懂了吗

```
public List GeValue() //进程
{
var resultA = factoryA.getResult(); //线程
var resultB = factoryB.getResult(); //线程
return ...
}
```

2016年6月23日 10:35 | # | 引用

阮小二 说：

我觉得比喻的挺好的。至于 线程是CPU的基本调度单位, 也能从工人的点去理解, 因为工厂里的基本单位也是人, 即使分成小组, 来调度, 也是人为单位的。虽然看完这篇博客中的讲解, 就是我现在还理解不了代码中的 多线程, 比如 张宴(赶集网架构师)的博客中 curl 抓取页面。请看到的同学 帮忙理解一下成么。QQ 508440335

2016年7月 4日 14:23 | # | 引用

11111 说：

好文章，我也是如此理解 进程和线程的关系的。

进程就是个完整的程序.可以单独运行一个exe 文件就是 一个进程。

线程就是进程中多个运行路线。

cpu控制这线程的运行。

2016年7月14日 16:04 | # | 引用

小蜜蜂 说：

看了原文和留言, 收获不少, 谢谢, 自己粗浅的总结如下:

(1)进程process：程序的一次执行(进程与进程之间有天然独立的内存空间), 可包括1个或多个线程。

(2)线程thread：CPU的基本调度单位。同一进程里的内存空间天然共享, 不同进程里的线程则不共享。内存线程间同步是为了防止竞争(就是说因同时修改, 而导致的数据不一致)。

(3)进程与线程的关系：线程包括进程, 进程封装在线程中。

2016年7月15日 19:52 | # | 引用

qwe 说：

> **引用jain的发言：**
>
> 搞这么复杂干嘛, 写段代码不就浅显易懂了吗
>
> ```
> public List GeValue()//进程
> {
> var resultA = factoryA.getResult(); //线程
> var resultB = factoryB.getResult(); //线程
> return ...
> }
> ```

你这个都是线程吧, 第一个算进程?

2016年8月10日 10:05 | # | 引用

林 说：

学到了, 谢谢

2016年8月20日 23:32 | # | 引用

黄亭亭 说：

阮老师真的是大牛, 您的博客页面很好看

2016年8月30日 23:33 | # | 引用

入水的鱼渴望海洋 说：

初接触, 感觉体会好深。电脑的硬件资源是一定的, 上层的软件系统就是共用这些资源完成不同的工作。

2016年9月 3日 14:31 | # | 引用

ice_bear 说:

我觉得很舒服, 这样的文章可以加深印象, 评论里很多人写出了更深刻更准确的定义, 但是这种定义就是走马观花, 看的时候觉得很有道理, 看完之后想想那个人写了什么, 头脑就一片空白了。喜欢博主的文章, 以后也会继续关注。

2016年9月 9日 11:27 | # | 引用

sunman721 说:

还真是不错!有点懂了!

2016年10月 4日 10:06 | # | 引用

杨洋 说:

一直没明白进程和线程, 有了阮老师的图片解说, 秒懂。原创知识一定要坚持下去, 请问有公众号吗? 想多向阮老师学习。

《沙海》, 闲暇之余的消遣读物, 一旦看上, 根本停不下来。htt://shahai.org

谁看谁知道。

2016年10月 4日 14:04 | # | 引用

冬瓜蔡 说:

思考下这种常见:如果服务器是多核, 但是只有一个任务, 并且任务是多线程的。那么这个任务是运行在多处理器上还是只占用了其中一个CPU

2016年10月11日 10:31 | # | 引用

whereami 说:

图画的不错

2017年1月10日 21:18 | # | 引用

李争 说:

http://blog.csdn.net/zheng548/article/details/54669908
我也写了关于线程和进程的一篇文章。欢迎大佬拍砖

2017年1月22日 20:18 | # | 引用

winfirm 说:

很好理解啊

2017年8月 4日 15:58 | # | 引用

Jerry 说:

互斥锁和信号量区别原来就在于资源个数, 搜嘎斯呢

2017年8月15日 21:01 | # | 引用

AaronHU 说:

@小蜜蜂:

(3)写反了吧

2017年8月28日 16:57 | # | 引用

学生 说:

这学期的课程 我也觉得线程才是基本单位 进程感觉是个没必要存在的概念 是这样吗

2017年9月12日 10:03 | # | 引用

菜鸟1号 说:

温故而知新

2017年9月20日 10:37 | # | 引用

樣 说:

> **引用屌丝男的发言:**
>
> 进程就是我们打开的一个快播播放器,线程就是快播里面的那些功能,如搜索、播放、暂停等。。

之前看了那么多例子, 一直有疑惑, 多线程要用锁来阻塞其他线程而防止数据混乱, 要用多线程的话有要开启多进程, 那为啥还要用多线程呢!看到了播放器, 恍然大悟

2018年5月18日 10:19 | # | 引用

kaidi 说:

> **引用阮一峰的发言:**
>
> 我完全同意你对进程和线程的定义。
>
> 这篇文章的比喻, 确实不是完全准确。但是, 进程和线程要找到现实生活中准确的对应, 恐怕很困难吧。
>
> 我的基本想法是:
>
> (1)进程是一个容器。
>
> (2)线程是容器中的工作单位。
>
> 车间—工人的比喻就是这样来的。

我觉得这个定义最贴切, 最初理解时, 也是很迷惑, 但是后来看到容器概念后, 有种恍然大悟的感觉.

线程, 进程, 容易弄混的是有部分人为对比因素,非要把两个概念抽象出相同点, 尤其是在单进程 单线程 程序中

2018年6月 6日 16:57 | # | 引用

小龙 说:

6中车间的空间是工人们共享的, 比如许多房间是每个工人都可以进出的。这象征一个进程的内存空间是共享的, 每个线程 此处应该是进程吧

2018年9月 5日 15:21 | # | 引用

pcli 说:

搞清楚进程 线程是干什么的, 解决了什么问题? 这样才能真正理解

2018年9月20日 19:43 | # | 引用

午後のお茶で一杯 说：

> **引用jain的发言：**
>
> 搞这么复杂干嘛, 写段代码不就浅显易懂了吗
>
> ```
> public List GeValue()//进程
> {
> var resultA = factoryA.getResult(); //线程
> var resultB = factoryB.getResult(); //线程
> return ...
> }
> ```

锁呢?交互呢?呵呵

2018年9月26日 13:34 | # | 引用

Troublemaker 说：

配图 里有《怪兽电力公司》哎 哈哈 形容的很贴切

2018年10月18日 16:39 | # | 引用

suxiongit 说：

@小蜜蜂：

你第3点描述进程与线程的关系, 是不是搞混?

2018年12月 6日 15:31 | # | 引用

liao 说：

> **引用mousycoder的发言：**
>
> @herodot：
>
> 线程其实是轻量级进程(Lightweight Process, LWP)，这个才是根本，进程才是CPU调度基本单位。
>
> 阮兄的比喻确实不够恰当，其实这些概念翻译过来，都变味了，不过你的评判大部分是对的，少部分还是有些问题。
>
> 详情可以看下：http://mousycoder.com/the-pragmatic-sa-13/

线程是一种轻量级的进程, 与进程相比, 线程给操作系统带来侧创建、维护、和管理的负担要轻, 意味着线程的代价或开销比较小。但是线程才是CPU调度基本单位。进程是资源分配的最小单位, 但是那是在没有引入线程以前, 在那之前, 进程既是最小资源分配单位, 也是最小调度单位。后来引入了线程, 最小调度单位就是线程,

2020年1月 3日 17:05 | # | 引用

Kenneth 说：

浅显易懂, 赞。

2020年11月25日 17:31 | # | 引用

淹不死的鱼 说：

深入浅出, 不错撒

2021年3月10日 10:33 | # | 引用

我要发表看法

您的留言（HTML标签部分可用）

您的大名：

«-必填

电子邮件：

«-必填, 不公开

个人网址：

«-我信任你, 不会填写广告链接

记住个人信息?

发表 «- 点击按钮

Weibo | Twitter | GitHub

Email: yifeng.ruan@gmail.com