DOI:10.19695/j.cnki.cn12-1369.2010.10.105

操作系统中的进程、线程与 Java 的多线程

虎治勤
(宁夏广播电视大学 宁夏银川 750002)

摘 要:本文从进程、线程、多线程的基本概念和属性出发,对进程与线程、Java与多线程的关系进行详细地论述,并对Thread类在Java程序中的应用、线程的创建和运行等提出示例。

关键词:进程 线程 Java的多线程;
中图分类号:TP311.1 文献标识码:A 文章编号:1007-9416(2010)10-0146-02

OS of the Process、Thread and Javamultithread

Hu Zhi qin
Ningxia TV University, Yinchuan 750002, China

Abstract: This paper reports a study on detailed expositions about the relationship among the process , threading and multi-thread ed java , analyzing from the basic concept of the process, threading and multi-threading , demonstrations are also offered to show th e applications of thread in the procedures Java、creation of threading and running.

Key words: Process Threading Javamultithread

1 进程与线程

操作系统中进程是指特定的代码序列在指定数据集合上的一次执行活动,是指并行程序的一次执行过程,在Windows95中,就是一个EXE文件的执行过程。是一个动态概念,具有动态属性,每一个进程都是由内核对象和地址空间所组成的,内核对象可以让系统在其内存放有关进程的统计信息并使系统能够以此来管理进程,而地址空间则包括了所有程序模块的代码和数据以及线程堆栈、堆分配空间等动态分配的空间。

通俗点讲,进程就是正在计算机上运行的可执行文件针对特定的输入数据的一个实例,从此意义上讲,进程应包含三部分内容,即:进程=PCB+程序段+数据(PCB:进程控制块),同一个可执行程序文件如果操作不同的输入数据就是两个不同的进程。

进程理解为一个线程容器,线程不能独立存在,它必须隶属于某个进程,而进程也至少拥有一个线程,如果一个进程的所有线程都结束了那么进程也就结束了。

所谓线程是指由进程进一步派生出来的一组代码(指令组)的执行过程。一个进程可以产生多个线程,这些线程都共享该进程的地址空间,它们可以并行、异步执行。在80年代后的现代操作系统中引入的线程,使之仅做为调度和分派的基本单位,但不拥有资源(仅拥有一点其运行中必不可少的资源:程序计数器、一组寄存器、栈),以便轻装运行,又不频繁对之调度、切换。采用线程最主要的好处是:使同一个程序能有几个并行执行的路径,提高并行执行程度,从而进一步提高执行速度;线程需要的系统开销比进程要小。应该说明的是,在Windows95中,“多任务”是基于线程而不是基于进程。

线程是允许进行并行计算的一个抽象概念,在另一个线程完成计算任务的同时,一个线程可以对图像进行更新,二个线程可以同时处理同一个进程发出的二个网络请求。

线程是进程调度的基本单位,负责执行在进程地址空间内的代码,线程是进程

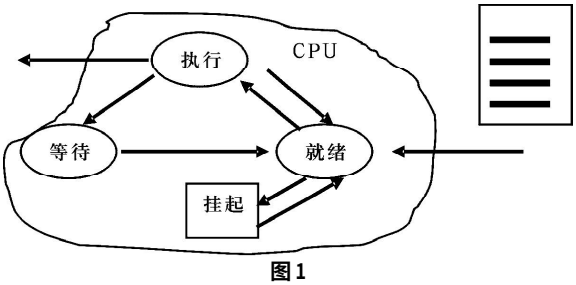

图1

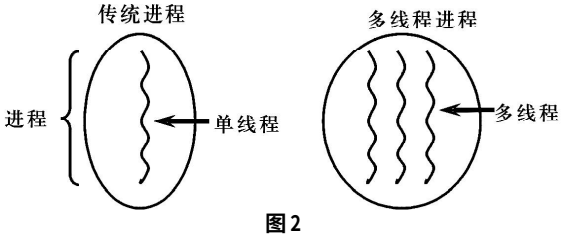

图2

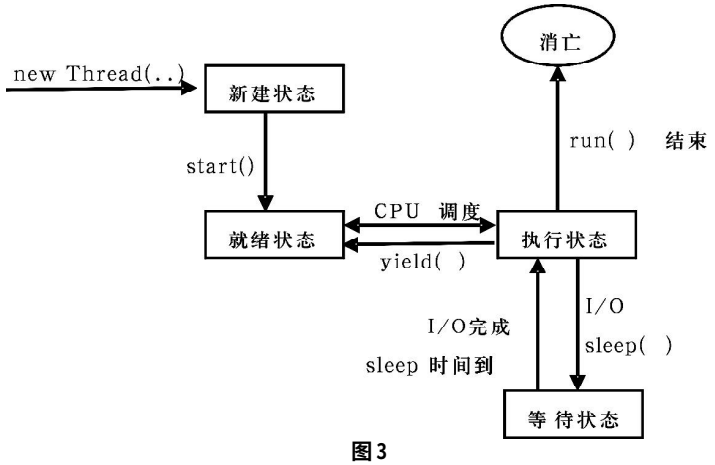

图3

的一条执行路径,它包含独立的堆栈和CPU寄存器状态,每个线程共享所有的进程资源,包括打开的文件、信号标识及动态分配的内存等等。由此可见,传统的进程可称之为重型进程 ,线程可称之为轻型进程。

进程有三种基本状态:就绪、阻塞、执行。线程同样也有同样三种基本状态。在现代操作系统中,进程是分配单元,而线程是执行单元(如图1)。

2 进程与JAVA线程的区别

应用程序在执行过程中存在一个内存空间的初始入口点地址、一个程序执行过程中的代码执行序列以及用于标识进程结束的内存出口点地址,在进程执行过程中的每一时间点均有唯一的处理器指令与内存单元地址相对应。

Java语言中定义的线程(Thread)同样包括一个内存入口点地址、一个出口点地址以及能够顺序执行的代码序列。但是进程与线程的重要区别在于线程不能够单独执行,它必须运行在处于活动状态的应用程序进程中,因此可以定义线程是程序内部的具有并发性的顺序代码流。

Unix操作系统和Microsoft Windows操作系统支持多用户、多进程的并发执行,而Java语言支持应用程序进程内部的多个执行线程的并发执行。多线程的意义在于一个应用程序的多个逻辑单元可以并发地执行。但是多线程并不意味着多个用户进程在执行,操作系统也不把每个线程作为独立的进程来分配独立的系统资源。进程可以创建其子进程,子进程与父进程拥有不同的可执行代码和数据内存空间。而在用于代表应用程序的进程中多个线程共享数据内存空间,但保持每个线程拥有独立的执行堆栈和程序执行上下文(Context)。

基于上述区别,线程也可以称为轻型进程 (Light Weight Process,LWP)。不同线程间允许任务协作和数据交换,使得在计算机系统资源消耗等方面非常廉价。

线程需要操作系统的支持,不是所有类型的计算机都支持多线程应用程序。Java程序设计语言将线程支持与语言运行环境结合在一起,提供了多任务并发执行的能力。这就好比一个人在处理家务的过程中,将衣服放到洗衣机中自动洗涤后将大米放在电饭锅里,然后开始做菜。等菜做好了,饭熟了同时衣服也洗好了。

需要注意的是,在应用程序中使用多线程不会增加 CPU 的数据处理能力。只有在多CPU 的计算机或者在网络计算体系结构下,将Java程序划分为多个并发执行线程后,同时启动多个线程运行,使不同的线程运行在基于不同处理器的Java虚拟机中,才能提高应用程序的执行效率。

3 进程与多线程

多线程是指同时存在几个执行体,按几条不同的执行线索共同工作的情况。实现单个进程中的并发计算。各线程间共享进程空间的数据,并利用这些共享单元来实现数据交换、实时通信与必要的同步操作。多线程的程序能更好地表述和解决现实世界的具体问题,是计算机应用开发和程序设计的一个必然发展趋势(如图2)。

4 Java与多线程

Java语言的一个重要功能特点就是内置对多线程的支持,它使得编程人员可以很方便地开发出具有多线程功能,能同时处理多个任务的功能强大的应用程序。Java的所有类都是在多线程的思想下定义的,Java利用线程使整个系统成为异步。每个Java程序都有一个隐含的主线程:

application main 方法

Applet小程序,主线程指挥浏览器加载并执行java小程序。

4.1 Thread类在Java程序中的应用

Thread类综合了Java程序中一个线程需要拥有的属性和方法,当生成一个Thread类的对象后,一个新的线程诞生了。每个线程都是通过目标对象的方法run( )来完成其操作的。方法run( ) 称为线程体。提供线程体的目标对象是在初始化一个线程时指明的。

任何实现了Runnable接口(实现run()方法)的类实例都可以作为线程的目标对象。如:

```
public class ThreadTest {
public static void main(String[] args){
Job1 j = new Job1( );
Thread t1 = new Thread(j) ;
t1.start( ) ;
} }
class Job1 implements Runnable {
int I;
public void run() {
while (i<50) System.out.println(i++) ; }
}
```

线程在执行过程中存在下面状态转换:(如图3)

4.2 线程的创建和运行

Java在Java.lang.Thread和java.lang.Runnable类中提供了大部分的线程功能。创建一个线程非常简单,就是扩展Thread类,并调用start()。通过创建一个执行Runnable()的类,并将该类作为参数传递给Thread(),也可以定义一个线程。下面是一个简单的Java程序实例,其中有2个线程同时在从1数到5,并将结果打印出来。

```
public class ThreadingExample
extends Object {
public static void main( String args[] ) {
Thread[] threads = new Thread[2];
for( int count=1;count<=threads.length;count++ ) {
threads[count] = new Thread( new Runnable( ) {
public void run( ) {
count( );
} } ;
threads[count].start( );
} }
public static void count( ) {
for( int count=1;count<=5;count++ )
System.out.print( count + " " );
} }
```

4.3 线程的使用

Java中存在许多编程人员希望能够对线程使用的标准操作:例如,测试线程是否存在、加入一个线程直到它死亡、杀死一个线程等。

结语

从传统操作系统中的进程到现代操作系统中的线程,再到多线程的核心目标是更好的提高多道环境下程序并行执行的程度,从而更好的提高系统吞吐量和系统处理能力。当然在一定程度上同时也增加了系统的时空开销。

参考文献

[1] 汤子赢等著,《计算机操作系统》,西安:西安电子科技大学出版社,1999.390～414.

[2] [美]Robin Rark著,前导工作室译,《UNIX技术大全》.北京:机械工业出版社,1998.

[3] 孟庆昌等著《UNIX教程》(修订本)西安:电子工业出版社出版 .2000.141～175.

[4] [美]Dave Taylor著,李际译,《UNIX教程》(第三版),北京:人民邮电出版社,2002.