文章
姚 鑫 · 六月 26, 2022 阅读大约需 7 分钟

第十一章 信号（一）- 概念

第十一章 信号（一） - 概念

背景

维基百科对信号量有这样的定义：“ 在计算机科学中，特别是在操作系统中，信号量是一种变量或抽象数据类型，用于控制多个进程在并行编程或多用户环境中对公共资源的访问。 ”信号量不同于互斥体（或锁）。互斥锁最常用于管理竞争进程对单个资源的访问。当一个资源有多个相同的副本并且这些副本中的每一个都可以由单独的进程同时使用时，就会使用信号量。

考虑一个办公用品商店。它可能有几台复印机供其客户使用，但每台复印机一次只能由一个客户使用。为了控制这一点，有一组键可以启用机器并记录使用情况。当客户想要复印文件时，他们向职员索取钥匙，使用机器，然后归还钥匙，并支付使用费。如果所有机器都在使用，客户必须等到钥匙归还。保存键的位置用作信号量。

该示例可以进一步推广到包括不同类型的复印机，也许可以通过它们可以制作的副本的大小来区分。在这种情况下，将有多个信号量，如果复制者在复制的大小上有任何重叠，那么希望复制共同大小的客户将有两个资源可供提取。

介绍

信号量是共享对象，用于在进程之间提供快速、高效的通信。每个信号量都是类 %SYSTEM.Semaphore 的一个实例。信号量可以建模为一个共享变量，它包含一个 64 位非负整数。信号量上的操作在共享它的所有进程中以同步的方式更改变量的值。按照惯例，值的变化会在共享信号量的进程之间传递信息。

尽管信号量和锁似乎有很多共同点，但使用信号量还是有一些优势的。其中最主要的事实是，当信号量被授予时，信号量会导致发送通知。因此使用信号量的进程不需要花费处理器周期或复杂的应用程序逻辑轮询锁以检查它是否已被释放。此外，信号量在 ECP 连接上透明地工作，并且比在相同情况下检查锁所需的交换效率更高。

信号概述

信号名称

信号量由它们的名称标识，这些名称在创建信号量时作为 ObjectScript 字符串提供。为信号量指定的名称应符合局部和全局变量的规则。信号量名称只是为了将其与所有其他现有信号量区分开来。有效名称字符串的示例有：SlotsOpen, J(3), ^pendingRequest(""j"")。

通常，信号量存储在创建该信号量的实例上，并且对该实例上的所有进程可见。但是，当信号量名称看起来像全局变量的名称时，信号量存储在映射全局变量(包括下标)的系统上。这允许这样的信号量对在ECP系统的实例上运行的所有进程可见。

注意：信号量的名称是一个字符串，但在运行时可以由诸如"^" _BaseName _"(" _(1 + 2) _ ")"这样的表达式构造。如果BaseName包含字符串 “ prters ”，则信号量名称为^prters(3)。

信号量值

信号量值存储为63位无符号整数，因此信号量值始终大于或等于零。

最大的63位无符号整数是9,223,372,036,854,775,807((2**63)-1)。信号量不能超过此值递增，也不能递减到零以下。

信号量实例和变量

信号量是派生自 %SYSTEM.Semaphore 的类的实例。一旦信号量被创建和初始化，它的 OREF 通常存储在一个 Objectscript 变量中，因此它可以用于其他操作，作为参数传递，并最终被删除。尽管包含对信号量的引用的变量的名称不必与信号量的名称相对应，但良好的编程习惯表明存在某种关系。

像所有对非持久数据的对象引用一样，当最后一个信号量引用被回收时，底层信号量也被删除。

信号量操作

基础

信号量操作可以分为两大类：直接操作信号量的操作和等待其他进程操作信号量的操作。第一组包括：
- Create – 创建一个新的信号量实例并初始化它以供使用
- Open ——访问并初始化现有的信号量
- Delete - 使任何知道信号量的进程无法使用它
- Increment - 将指定量添加到信号量的值
- Decrement ——如果信号量值为零，则操作等待它变为正值。当信号量为正时，从信号量中减去递减量（或信号量的值，以较小者为准）。
- GetValue – 返回信号量的当前值
- SetValue – 将信号量的当前值设置为提供的非负值

管理多个信号量

第二组操作涉及将信号量列表作为一个组进行管理，并等待每个 on 待处理的操作完成：
- AddToWaitMany – 将给定的信号量操作添加到等待列表
- RemoveFromWaitMany – 从等待列表中删除指定的信号量操作
- WaitMany – 等待等待列表中的所有信号量完成各自的操作。此操作可能会超时。

等候队列

每个使用 WaitMany 协调多个信号量的进程将信号量减量操作请求保存在一个内部列表中。对列表的操作处理如下：

- 当调用AddToWaitMany方法在列表中放置递减操作时，系统会尝试在此时执行递减。如果信号量值非零，则递减成功。减去的量是信号量的值和请求的量中较小的一个。任何大于信号量的值的请求都会被忘记。
- 如果在将请求添加到列表时信号量的值为零，则不执行任何操作，并且该请求被视为挂起。在未来的某个时间，如果目标信号量变为非零，将选择其中一个进程，其操作引用该信号量并执行其递减操作。如果该操作的结果是信号量仍然具有非零值，则将重复该过程，直到没有进一步的请求，或者信号量的值变为零。
- 当进程调用WaitMany方法时，会检查等待列表中的每个操作。对于满足的请求，调用目标信号量的WaitComplete方法，然后从等待列表中删除该请求。当它处理完所有满足的请求时，它向调用方返回此类请求的数量；如果超过等待超时，则返回零。待处理和未满足的请求仍在等待名单上。
- 由于信号量的异步性质，在调用WaitMany期间，等待列表上的某个挂起请求可能会得到满足。在这次WaitMany调用期间是否计入这个现已满足的请求，或者它是否将计入后续调用，目前尚不确定。
- 将OREF保存到信号量的进程也可能删除它。在这种情况下，会发生什么取决于请求的操作是否得到满足。
    - 如果信号量是已满足的操作的目标，则将该请求标记为已将信号量递减零。无法调用WaitComplete方法，因为信号量不存在，但该请求被视为已在WaitMany返回的值中得到满足。
    - 如果请求仍处于挂起状态，则只需将其从等待列表中删除。

回调

信号量的实例继承一个抽象方法WaitComplete，用户需要实现该方法。在处理等待列表上满足的请求时，WaitMany对等待列表中的每个信号量调用WaitComplete方法，并将信号量递减的数量作为参数传递。当WaitComplete返回时

，WaitMany将从等待列表中删除该请求。

其他考虑事项

同一等待列表上有多个递减请求

在同一等待列表中多次请求递减同一信号量并不是错误的。添加的请求处理方式如下：
- 如果第一个请求不被满足，则将第二个请求的减少量与第一个请求的减少量相加。
- 如果已满足第一个请求(全部或部分)，则正常处理第二个请求。也就是说，如果信号量值非零，则完全或部分满足递减请求。但是，减去的实际数量与第一个请求获得的数量相加。

在调用WaitMany之前，不会通过回调报告递减量，因此对同一信号量的多个请求看起来就像是发出了一个组合请求。这会导致以下情况：

1. 信号量A设置为0。
2. 对 A 的减量请求 4。
3. 对 A 的减量请求为 1；新的减量 = 5。
4. 信号量 A 设置为 4。
5. 请求满足； 4 授予。
6. call waitmany ；等待完成A报告。
7. 信号量A设置为1。
8. A的减量请求，共3个。
9. 请求满足； 1 授予。
10. 对 A 的减量请求 4。
11. call WaitMany； WaitCompleted for A 报告 1。
12. 信号量 A 设置为 1。
13. 对 A 的减量请求为 3。
14. 请求满足； 1 授予。
15. 对 A 的减量请求 4。
16. 信号量 A 设置为 5。
17. 请求满足； 4 授予。
18. 调用WaitMany； WaitCompleted for A 报告 5；信号量 A 值 = 1。

信号量删除

信号量没有所有者，并且它们不像对象实例那样被引用计数。任何可以打开信号量的进程都可以将其删除。

当一个信号量被删除时，

1. 如果任何等待列表中存在该信号量的挂起递减，则调用 WaitCompleted 回调，递减值为零。
2. 它将从映射的系统（本地或远程）中删除。
3. 任何其他进程进一步尝试访问它都会失败，并出现 <INVALID SEMAPHORE>错误。

工作终止和等待列表

进程终止时，它的等待列表被释放。保留在等待列表中但尚未由 WaitMany 处理的任何满足的递减请求都将被清除。它们各自的递减量不会添加回它们递减的信号量。等待列表中的任何未满足的请求都将被简单地删除。

信号量和 ECP

对于 ECP 系统上的信号量，信号量上的操作按照请求到达持有信号量的系统的顺序进行排序。每个操作都保证在下一个操作开始之前完成。远程信号量保证以下条件：

- 信号量增加和减少将在 SET 和 KILL 之后发生。
- 当一个信号量被 SET、递增或递减时，ECP 数据缓存与服务器上的后续 SET、递增或递减是一致的。

因为信号量不是持久的，所以在服务中断的情况下，ECP
系统上跨服务器的未决信号量操作是不可恢复的。由于服务器或网络故障导致 ECP 中断后，应用服务器上的信号量将被删除，数据服务器上的待处理请求也将被删除。应用程序有责任检测这种情况并在正确状态下重新创建所需的信号量。

#SQL #Caché

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B8%80%E7%AB%A0-%E4%BF%A1%E5%8F%B7%EF%BC%88%E4%B8%80%EF%BC%89-%E6%A6%82%E5%BF%B5