Linux中的冲突问题及其应对策略 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022_Linux_E4_B8_AD_E7_9A_c103_144324.htm Linux系统的稳定性成为很多评论家们反对冲突不断的Windows系统的一个很好的武器。然而，Linux系统的冲突问题虽然比较少，但是一旦在意想不到的情况下出现，也很容易使人们陷入困境。学习一些常用手段来预防这些这些冲突问题的发生是十分重要的，它可以帮助Linux的系统管理员们避免那些困境情况的出现。 在本站的采访中，Mark Wilding和Dan Behman对于Linux系统冲突问题的预防及修复提供了一种比较简捷明了的方法。他们两人共同出版了一本新书《Self-Service Linux: Mastering the Art of Problem Determination》。 一般认为，Linux服务器系统是不存在冲突的，然而些许时候该系统的冲突、停滞问题的确存在。对于应用软件层面的冲突或者停滞问题，与内核层面有何不同呢? Mark Wilding: 应用软件层面的冲突或者停滞问题只是限制于一个特定的线程或者进程。这种冲突或者停滞问题不会导致在该相同系统上运行的其他线程或者进程的冲突或者停滞。然而，如果是在内核层面上发生，那么它将影响到该系统中运行的所有进程。 系统的冲突和停滞，这二者有什么区别呢? Dan Behman: 在任何一个层面，冲突和停滞这二者的属性基本上是相同的。停滞发生在进程或线程受阻的时候，此时是由于某种锁定或者某些硬件资源的繁忙，从而该进程或线程不得不等待。等待某些锁定或资源的情况是经常发生的，但是只有当这种锁定或者资源最终无法实现的时候，才会引起系统停滞。 还有很重要的一点需要注意的是，停滞

问题有时候可以提早的诊断出来。我的意思是，例如，某种资源的某个特定的时刻非常的繁忙，这是需要这种资源的进程或线程就需要等待非常长的一段时间，直至该资源空闲下来。用户经常不了解资源的这种繁忙状况，而只看到该进程在等待，所以他就认为系统发生了停滞，但实际上此时系统仍然在按照既定的工作流程进行，只是速度比较慢。 而系统的冲突问题与上述的停滞是不同的，它主要是由于某种不可知的硬件或软件错误而导致的。当这种错误发生时，特殊的错误处理程序将很可能会调用那些诊断信息及报告，从而有希望能够追踪到这种错误的原因。 冲突问题可以看作是某种致命的问题，它需要完结后才能够进行分析。而停滞问题可以看作是实时的问题，它可以即时的进行分析、解决。 我知道Linux有一个最大的优势就是在于它的源代码的开放性.除此之外，还有别的原因致使Linux比其他操作系统的冲突问题容易解决吗? Behman: 伴随着这种源代码的开放性，在Linux系统的每一个层面都有着相当多的参阅文件。同时，既然源代码是开放的，那么它的开发团队也同样是开放的。这样以来，你就可以把所遇到的问题向Linux内核的开发者求助，当然包括最初的那些开发人员，甚至Linus Torvalds本人，而这所有的求助程序也仅仅是发送一封电子邮件就可以了。而据我所知，Linux的这种能力是那些不开放源码的操作系统所缺少的。 处理停滞问题有那些困难和挑战呢? Wilding: 一个应用软件的停滞问题是有着多种原因的，也包括那些可能由于内核空间的问题所引起的停滞。这意味着有时候这些问题不是开发者所能够控制的。但是这正是Linux的优势所在。所有的源代码都是开放的，所以如果你遇到了某种进程的内核阻滞状况

，那么就可以联系其源代码，从而查看该进程在内核中是如何作用的。然而，在大部分情况下，是没有必要进行这么深层次研究的。为了探究进程停滞的原因到底何在?应用软件的开发者需要认真的研究这些软件层面的状况及证据(例如，堆栈的路径等)。 对于用户或者维护人员而言，他们一般不会了解应用软件的具体工作程序，也不会没有能力进入到源代码层面进行测试，这是遇到系统停滞问题可以灵活的进行处理。例如，在某种情况下，进程A在等待进程B结束后所释放的资源，而进程B又在等待进程A占有的资源。这就是所谓的“死锁”，这也正是复杂应用软件中经常出现的问题，可以作为停滞问题的一种诊断方案。 如果你不清楚进程A及进程B的具体等待原因，那么你甚至不需要明白这到底是不是“死锁”的情况，而别无选择的关掉这两个进程后重新开启。正是这种类似情况，因此对于应用软件而言，进行完全的资源及上锁情况的跟踪是十分重要的，这可以帮助解决这种比较棘手的问题。 Behman: 关于停滞问题的另一个挑战在于，当停滞问题发生时，进程或者线程经常不知道它被停滞了或者何时将被停滞。这种情况和冲突问题是不同的，当冲突问题发生时，进程可以截取大部分的信号，并且信号处理程序可以添加到平台系统中来处理这些特殊情况，例如清理内存，堆栈跟踪等等。但是，当停滞问题发生时，这种特殊的处理程序虽然不是完全不可能进行，但是往往比较灵活，不太固定。 停滞问题发生时，往往去重新启动该系统或者应用软件。有一点需要切记的是，发生停滞问题时，诊断该问题的一些资料和证据经常被活动的内核及应用软件所捕获。如果你不收集这些重要的信息而立即重新启动，那么你永远不知道如

何去诊断这种问题，从而也就不可能阻止其将来的再次发生。对于一些特殊重要的环境而言，系统的稳定性及可靠性是与问题诊断及解决速度紧密相连。因此，需要坚持一种合理的思路，那就是“先收集错误信息，然后重新启动”。与冲突问题对比，当遇到停滞问题时，首先要做的事情是什么呢? Behman: 处理内核层面的停滞问题与处理应用软件层面的停滞是有着很大不同的。如果你问的是关于应用软件层面。当冲突问题发生时，有着一种称为“信号处理”的特殊功能来调用处理各种各样的信息，例如内存中的信息，堆栈的跟踪反馈等。所以一般情况下，遇到冲突问题时，首要的问题就是收集、整理、分析这些数据。而在停滞问题发生时，这种数据不会自动的收集，而这往往是一种人工的操作过程。收集停滞状态数据的两个关键点在于追踪输出结果以及堆栈的跟踪反馈。这种追踪输出结果的方式能够得出该进程作用情况的信息，因为它在一直的监视该进程.这些信息例如，该进程是否仍然在作用等等。而堆栈的跟踪反馈可以给出目前进程作用的源代码部分。这对于开发人员是非常重要的，这样以来他们就可以研究该进程停滞问题产生的原因。对于冲突及停滞问题，其最主要的原因是什么呢? Wilding: 对于冲突问题而言，我们可以把它的主要原因分为两种，一种是预防型的，另一种是错误处理型的。预防型冲突是内核或应用软件由于遇到了严峻的形势而产生冲突问题的情况。而软件意识到这种问题，并产生一种“自杀式”的方法以阻止错误的进一步发生，从而避免更加严重的问题出现。而对于错误处理型的冲突，它意味着内存中有着某些不合法的内容进入，几乎都是一些程序的错误。在这种情况下，硬件探测到这种应

用软件，然后发送信号去阻止该软件的进程。 对于停滞问题而言，也一般存在着两种原因状况。一种是进程或线程等待资源的情况，这不一定能否解决。而其他的进程或线程约束着该资源(例如，上锁)，这样该进程或线程在等待时，其还占据着资源，从而其他进程或线程也只能等待。一个例子就是某个进程对占据的重要资源上锁，而其自身在漫无目的接收因特网的信息。第二中比较常见的原因就是一种“ 依赖回路式 ”的等待，在此两个或者多个进程在互相等待它方的资源，从而陷入“ 死锁 ”。这种情况的解决方法可以是释放一个锁，或者在某个空间中共享内存等。 在这些冲突以及停滞的情形之下，管理者们有哪些基本的调查研究规则可以应用呢? Wilding: 一个最好的基本准则就是有组织的参加工作。很重要的一点就是把收集的数据有规则的放在一个明确的地点，这样将来能够很容易的找到。这对于那些同时遇到多个问题的情况尤为有用。 Behman: 另一个基本的准则就是要定量的收集数据，而不是定性的收集数据。例如，“ 昨天晚上6点，系统内存利用较低 ”，这是定性的观测。这在问题处理中的作用不大。关于该例子的定量的版本应该要收集并保存那些所有的输出的数据命令，以及其他一些相关的诊断命令。目的就是在于收集足够的数据，这样就可以尽可能的避免问题的再次发生.这就是“ 一次到位式 ”的方法，而不需要问题重复出现后，多次收集才能够得到比较完整的数据。 