The new way of ioctl()
ioctl()的新方法

[Posted January 18, 2005 by corbet]

The ioctl() system call has long been out of favor among the kernel developers, who see it as a completely uncontrolled entry point into the kernel. Given the vast number of applications which expect ioctl() to be present, however, it will not go away anytime soon. So it is worth the trouble to ensure that ioctl() calls are performed quickly and correctly - and that they do not unnecessarily impact the rest of the system.

ioctl()系统调用在内核开发者眼里早已不受待见，他们认为这是一个完全不受控制的内核入口。鉴于还有大量的应用程序希望使用 ioctl(),所以它不会很快的消失。所以它还是值得我们勉为其难地去保证 ioctl()系统调用快速正确地被执行，并且让它们不会对其系统的其它部分造成影响。

ioctl() is one of the remaining parts of the kernel which runs under the Big Kernel Lock (BKL). In the past, the usage of the BKL has made it possible for long-running ioctl() methods to create long latencies for unrelated processes. Recent changes, which have made BKL-covered code preemptible, have mitigated that problem somewhat. Even so, the desire to eventually get rid of the BKL altogether suggests that ioctl() should move out from under its protection.

ioctl() 是在内核中依然运行于大内核锁(BKL)保护下的部分之一。以前，BKL 的作用使得 ioctl()方法的运行期可能较长，即对其他进程产生较长的延时。最近的修改（在 BKL 保护下可抢占），在一定程度上缓解了这个问题。即便如此，鉴于最终完全剔除 BKL 的趋势，ioctl()还是应该脱离它的保护。

Simply removing the lock_kernel() call before calling ioctl() methods is not an option, however. Each one of those methods must first be audited to see what other locking may be necessary for it to run safely outside of the BKL. That is a huge job, one which would be hard to do in a single "flag day" operation. So a migration path must be provided. As of 2.6.11, that path will exist.

然而，简单地将调用 ioctl() 方法前的 lock_kernel()函数移除并不是一个好办法。每一个需要移除 BKL 的方法都必须好好地审查一下：是不是有必要使用其他的锁机制来替换，以使代码安全地运行于无 BLK 保护的情况下。这是一个巨大的工程，以至于不可能在短期内完成。所以必须提供一个过渡的方法。从 2.6.11 开始，这种方法将会出现。

The patch (by Michael s. Tsirkin) adds a new member to the file_operations structure:
这种方法 (by Michael s. Tsirkin)是向 file_operations 结构体中添加一个新的成员：

```
long (*unlocked_ioctl) (struct file *filp, unsigned int cmd, unsigned long arg);
```

If a driver or filesystem provides an unlocked_ioctl() method, it will be called in preference to the older ioctl(). The differences are that the inode argument is not provided (it's available as filp->f_dentry->d_inode) and the BKL is not taken prior to the call. All new code should be written with its own locking, and should use unlocked_ioctl(). Old code should be converted as time allows. For code which must run on multiple kernels, there is a new HAVE_UNLOCKED_IOCTL macro which can be tested to see if the newer method is available or not.

ioctl()的新方法

tekkamanninja@gmail.com

如果一个驱动或者文件系统提供一个 unlocked_ioctl() 方法，它将会在被调用时优先于老的 ioctl()。不同的是 inode 参数不再被提供（ 可通过 filp->f_dentry->d_inode 获得），且在调用之前不使用 BKL。所有新代码必须自己添加锁机制，并使用 unlocked_ioctl()。老的代码在时间允许的情况下必须转换过来。对于那些需要在不同的内核中运行的代码（译者注：如驱动模块），有一个 HAVE_UNLOCKED_IOCTL 宏可以用于测试新的方法是否可行。

Michael's patch adds one other operation:
Michael 的补丁添加了另一个 file_operations 中的成员：

```
long (*compat_ioctl) (struct file *filp, unsigned int cmd,unsigned long arg);
```

If this method exists, it will be called (without the BKL) whenever a 32-bit process calls ioctl() on a 64-bit system. It should then do whatever is required to convert the argument to native data types and carry out the request. If compat_ioctl() is not provided, the older conversion mechanism will be used, as before. The HAVE_COMPAT_IOCTL macro can be tested to see if this mechanism is available on any given kernel.

如果这个方法存在，它将会在一个 32 位的进程在 64 位系统上调用 ioctl() 时被使用（不使用 BKL）。它会将其中的参数做必要的转换，变为适当的数据类型，并完成函数的执行。如果 compat_ioctl() 没有实现，老的转换机制将会像原先一样运作。

The compat_ioctl() method will probably filter down into a few subsystems. Andi Kleen has posted patches adding new compat_ioctl() methods to the block_device_operations and scsi_host_template structures, for example, though those patches have not been merged as of this writing.

compat_ioctl()方法会适当的慢慢的进入其他的一些子系统。例如，Andi Kleen 已经发布了补丁，将新的 compat_ioctl() 方法添加到了 block_device_operations 和 scsi_host_template 结构体中。虽然这些补丁在写这篇文章时还没有被合并入主线内核。

(Log in to post comments)

more on compat_ioctl
更多关于 compat_ioctl

Posted Oct 23, 2005 14:19 UTC (Sun) by arnd (subscriber, #8866) [Link]

There are a few noteworthy points about compat_ioctl:

这里有一些关于 compat_ioctl 值得注意的地方：

If you are writing a new device driver that needs ioctl methods (which some might argue you should not do in the first place), make sure the data structure are compatible between 32 and 64 bit, so unlocked_ioctl and compat_ioctl can point to the same function. In particular, data structures containing must not contain fields that have different sizes (e.g. 'void *' or 'long') or need padding (e.g. 'long long' after 'int') on 64 bit systems.

如果你要编一些需要 ioctl 方法的新设备驱动（首先，有些人就会对你的这种做法不满），请确保数据结构是 32-bit 和 64-bit 兼容的，所以 unlocked_ioctl 和 compat_ioctl 可以指向同一个函数。 特殊情况下，数据结构不应该包含一些在 64-bit 系统中（译者注：与 32-bit 系统相比）有不同大小（比如，'void *'或'long'）或需要填充（例如 'int'之后的'long long')的区域。

As of 2.6.14, nobody has started converting the network layer to compat_ioctl, so the next person that needs new compatibility code for socket ioctls should add the infrastructure for that instead of adding on to fs/compat_ioctl.c.

在 2.6.14 内核中，还没有人开始转换网络层的 compat_ioctl，所以今后如果需要新的套接字中的 ioctls 兼容性代码，就应该为其添加基础代码，而不是往 fs/compat_ioctl.c 里添加代码.

While the fs/compat_ioctl.c infrastructure still exists, it is valid for compat_ioctl methods to return -ENOIOCTLCMD for anything they don't know. This is particularly useful for block or tty devices that have a lot of ioctl numbers common to all drivers. The vfs layer first calls ->compat_ioctl and if that does not exist or returns -ENOIOCTLCMD, it scans the list of known conversions between 32 and 64 bit ioctls and if it finds a valid conversion, it enters the native 64 bit ->unlocked_ioctl/->ioctl path.

而 fs/compat_ioctl.c 基础代码依然存在是为了在 compat_ioctl 方法返回 -ENOIOCTLCMD （或未实现 compat_ioctl）时使用的。这对于块设备或者 tty 设备这些包含大量 ioctl 操作的常见驱动来说是非常有用的。VFS 层先调用->compat_ioctl ，如果它不存在或者返回 -ENOIOCTLCMD，就会扫描已知的 32-bit 和 64-bit 之间的 ioctl 转换列表，如果找到了可用的转换，就会按以下的顺序执行：本地 64-bit 转换操作->unlocked_ioctl（或者->ioctl）