[linux内核空间和用户空间的是怎样区别的，如何交互，如何从用户空间进入内核空间](file:///E:\linux%E5%9F%BA%E7%A1%80%E7%A7%AF%E7%B4%AF\linux内核空间和用户空间的是怎样区别的，如何交互，如何从用户空间进入内核空间%20-%20bingqingsuimeng的专栏%20-%20博客频道%20-%20CSDN.NET_files\linux内核空间和用户空间的是怎样区别的，如何交互，如何从用户空间进入内核空间%20-%20bingqingsuimeng的专栏%20-%20博客频道%20-%20CSDN.NET.html)

标签： [linux内核](http://www.csdn.net/tag/linux%e5%86%85%e6%a0%b8)[linux](http://www.csdn.net/tag/linux)[任务](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1)[path](http://www.csdn.net/tag/path)[user](http://www.csdn.net/tag/user)[工作](http://www.csdn.net/tag/%e5%b7%a5%e4%bd%9c)

2012-08-30 12:55 9428人阅读 [评论](http://blog.csdn.net/bingqingsuimeng/article/details/7924756#comments)(5) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/bingqingsuimeng/article/details/7924756#report)

 分类：

linux驱动程序设计（49）

[linux](http://www.chinabyte.com/keyword/Linux/)驱动程序一般工作在内核空间，但也可以工作在用户空间。下面我们将详细解析，什么是内核空间，什么是用户空间，以及如何判断他们。

　　Linux简化了分段机制，使得虚拟地址与线性地址总是一致，因此，Linux的虚拟地址空间也为0~[4G](http://www.chinabyte.com/keyword/4G/).Linux内核将这4G字节的空间分为两部分。将最高的1G字节(从虚拟地址0xC0000000到0xFFFFFFFF)，供内核使用，称为"内核空间".而将较低的[3G](http://www.chinabyte.com/keyword/3G/)字节(从虚拟地址 0x00000000到0xBFFFFFFF)，供各个进程使用，称为"用户空间)。因为每个进程可以通过系统调用进入内核，因此，Linux内核由系统内的所有进程共享。于是，从具体进程的角度来看，每个进程可以拥有4G字节的虚拟空间。

　　Linux使用两级保护机制：0级供内核使用，3级供用户程序使用。从图中可以看出(这里无法表示图)，每个进程有各自的私有用户空间(0~3G)，这个空间对系统中的其他进程是不可见的。最高的1GB字节虚拟内核空间则为所有进程以及内核所共享。

　　内核空间中存放的是内核代码和数据，而进程的用户空间中存放的是用户程序的代码和数据。不管是内核空间还是用户空间，它们都处于虚拟空间中。

　　虽然内核空间占据了每个虚拟空间中的最高1GB字节，但映射到物理内存却总是从最低地址(0x00000000)开始。对内核空间来说，其地址映射是很简单的线性映射，0xC0000000就是物理地址与线性地址之间的位移量，在Linux代码中就叫做PAGE\_OFFSET.

　　内核空间和用户空间之间如何进行通讯？

　　内核空间和用户空间一般通过系统调用进行[通信](http://telecom.chinabyte.com/)。

　　如何判断一个驱动是用户模式驱动还是内核模式驱动？ 判断的标准是什么？

　　用户空间模式的驱动一般通过系统调用来完成对硬件的访问，如通过系统调用将驱动的io空间映射到用户空间等。因此，主要的判断依据就是系统调用。

　　内核空间和用户空间上不同太多了，说不完，比如用户态的链表和内核链表不一样;用户态用printf,内核态用printk;用户态每个应用程序空间是虚拟的，相对独立的，内核态中却不是独立的，所以编程要非常小心。等等。

　　还有用户态和内核态程序通讯的方法很多，不单单是系统调用，实际上系统调用是个不好的选择，因为需要系统调用号，这个需要统一分配。

　　可以通过ioctl、sysfs、proc等来完成。

在进行设备驱动程序，内核功能模块等系统级开发时，通常需要在内核和用户程序之间交换信息。Linux提供了多种方法可以用来完成这些任务。本文总结了各种常用的信息交换方法，并用简单的例子演示这些方法各自的特点及用法。其中有大家非常熟悉的方法，也有特殊条件下方可使用的手段。通过对比明确这些方法，可以加深我们对Linux内核的认识，更重要的是，可以让我们更熟练驾御linux内核级的应用开发技术。

**内核空间(kernel-space) VS 用户空间(user-space)**

作为一个Linux开发者，首先应该清楚内核空间和用户空间的区别。关于这个话题，已经有很多相关资料，我们在这里简单描述如下：

现代的计算机体系结构中存储管理通常都包含保护机制。提供保护的目的，是要避免系统中的一个任务访问属于另外的或属于[**操作系统**](http://lib.csdn.net/base/operatingsystem)的存储区域。如在IntelX86体系中，就提供了特权级这种保护机制，通过特权级别的区别来限制对存储区域的访问。 基于这种构架，Linux操作系统对自身进行了划分：一部分核心软件独立于普通应用程序，运行在较高的特权级别上，（Linux使用Intel体系的特权级3来运行内核。）它们驻留在被保护的内存空间上，拥有访问硬件设备的所有权限，Linux将此称为内核空间。

相对的，其它部分被作为应用程序在用户空间执行。它们只能看到允许它们使用的部分系统资源，并且不能使用某些特定的系统功能，不能直接访问硬件，不能直接访问内核空间，当然还有其他一些具体的使用限制。（Linux使用Intel体系的特权级0来运行用户程序。）

从安全角度讲将用户空间和内核空间置于这种非对称访问机制下是很有效的，它能抵御恶意用户的窥探，也能防止质量低劣的用户程序的侵害，从而使系统运行得更稳定可靠。但是，如果像这样完全不允许用户程序访问和使用内核空间的资源，那么我们的系统就无法提供任何有意义的功能了。为了方便用户程序使用在内核空间才能完全控制的资源，而又不违反上述的特权规定，从硬件体系结构本身到操作系统，都定义了标准的访问界面。关于X86系统的细节，请查阅参考资料1

一般的硬件体系机构都提供一种“门”机制。“门”的含义是指在发生了特定事件的时候低特权的应用程序可以通过这些“门”进入高特权的内核空间。对于IntelX86体系来说，Linux操作系统正是利用了“系统门”这个硬件界面（通过调用int $0x80机器指令)，构造了形形色色的系统调用作为软件界面，为应用程序从用户态陷入到内核态提供了通道。通过“系统调用”使用“系统门”并不需要特别的权限，但陷入到内核的具体位置却不是随意的，这个位置由“系统调用”来指定，有这样的限制才能保证内核安全无虞。我们可以形象地描述这种机制：作为一个游客，你可以买票要求进入野生动物园，但你必须老老实实的坐在观光车上，按照规定的路线观光游览。当然，不准下车，因为那样太危险，不是让你丢掉小命，就是让你吓坏了野生动物。

出于效率和代码大小的考虑，内核程序不能使用标准库函数（当然还有其它的顾虑，详细原因请查阅参考资料2）因此内核开发不如用户程序开发那么方便。

**内核空间和用户空间的相互作用**

现在，越来越多的应用程序需要编写内核级和用户级的程序来一起完成具体的任务，通常采用以下模式：首先，编写内核服务程序利用内核空间提供的权限和服务来接收、处理和缓存数据；然后编写用户程序来和先前完成的内核服务程序交互，具体来说，可以利用用户程序来配置内核服务程序的参数，提取内核服务程序提供的数据，当然，也可以向内核服务程序输入待处理数据。

比较典型的应用包括: Netfilter(内核服务程序：防火墙)VS Iptable（用户级程序：规则设置程序）；IPSEC(内核服务程序：VPN协议部分)VS IKE(用户级程序：vpn密钥协商处理)；当然还包括大量的设备驱动程序及相应的应用软件。这些应用都是由内核级和用户级程序通过相互交换信息来一起完成特定任务的。

**信息交互方法**

用户程序和内核的信息交换是双向的，也就是说既可以主动从用户空间向内核空间发送信息，也可以从内核空间向用户空间提交数据。当然，用户程序也可以主动地从内核提取数据。下面我们就针对内核和用户交互数据的方法做一总结、归纳。

信息交互按信息传输发起方可以分为用户向内核传送/提取数据和内核向用户空间提交请求两大类，先来说说：

**由用户级程序主动发起的信息交互。**