从socket can中断到netlink用户态内核态通信

1. Linux中的进程间的通信机制源自于Unix平台上的进程通信机制。Unix的两大分支AT&T Unix和BSD Unix在进程通信实现机制上的各有所不同，前者形成了运行在单个计算机上的System V IPC，后者则实现了基于socket的进程间通信机制。同时linux也遵循IEEE制定的posix IPC标准，在三者的基础上实现以下几种主要的IPC机制：管道(Pipe)和命名管道(Name Pipe)，信号(Signal)，消息队列(Message queue)，共享内存(Shared Memory)，信号量(Semaphore)，套接字(Socket)。通过这些IPC机制，用户空间进程之间可以完成互相通信。而为了完成内核空间和用户空间之间的通信，linux提供了基于socket的Netlink通信机制，可以实现内核与用户空间数据的及时交换。

2. linux操作系统中当cpu处于内核状态时，可以分为有用户上下文的状态和执行硬件、软件中断两种。其中当处于有用户上下文时，由于内核态和用户态的内存映射机制不同，不可直接将本地变量传给用户态的内存区；处于硬件、软件中断时，无法直接向用户内存去传递数据，代码执行不可中断。   
他们都无法直接在内核态和用户态之间使用，原因如下：

* + 管道   
    局限于父子进程间通信。
  + 消息队列   
    在软、硬中断中无法无阻塞地接收数据
  + 信号量   
    虽然原理一样，但内核空间和用户空间的信号量是两套完全的系统，所以信号量不能用于内核空间和用户空间信息交互
  + 内存共享   
    需要信号量辅助
  + socket套接字   
    在硬、软中断中无法无阻塞地接收数据

目前linux提供了

3. 解决内核态和用户态通信机制

处于有用户上下文时，可以使用linux提供的copy\_to\_user()和copy\_from\_user()函数完成，但由于这两个函数可能阻塞，因此不能在硬件、软件中断的过程中使用。

linux将内存空间分为两部分，将最高字节的一部分，供内核使用，称为“内核空间”，而较低字节的一部分，则为“用户空间” 处理器状态：

a. 内核态，运行于进程上下文，内核代表进程运行于内核空间；

b. 内核态，运行于中断上下文，包括硬中断和软中断；

c. 用户态，运行于用户空间。

处于硬、软件中断时，主要分为两类：

（1）可以通过linux提供的spinlock自旋锁实现内核线程和中断过程的同步，由于内核线程运行在有上下文的进程中，因此可以在内核线程中使用套接字或消息队列来取得用户空间的数据，然后再将数据通过临界区传递给中断过程。

（2）通过netlink机制实现。netlink套接字的通信依据是一个对应于进程的标识，一般定为该进程的ID。netlink通信最大的特点是对中断过程的支持，它在内核空间接收用户空间数据时不再需要用户自动启动一个内核线程，而是通过另一个软中断调用用户事先指定的接收函数。通过软中断而不是自行启动内核线程保证了数据传输的及时性。

AF\_NETLINK 与PF\_NETLINK