# 一、应用程序和驱动的交互原理

1、驱动就是获取外设、或者传感器数据，控制外设。数据会提交给应用程序。Linux驱动编译既要编写一个驱动，还要我们编写一个简单的测试应用程序，APP。单片机下驱动和应用都是放到一个文件里面，也就是杂糅到一起。Linux下驱动和应用是完全分开的。

用户空间(用户态)和内核空间(内核态)：

Linux操作系统内核和驱动程序运行在内核空间、应用程序运行在用户空间。

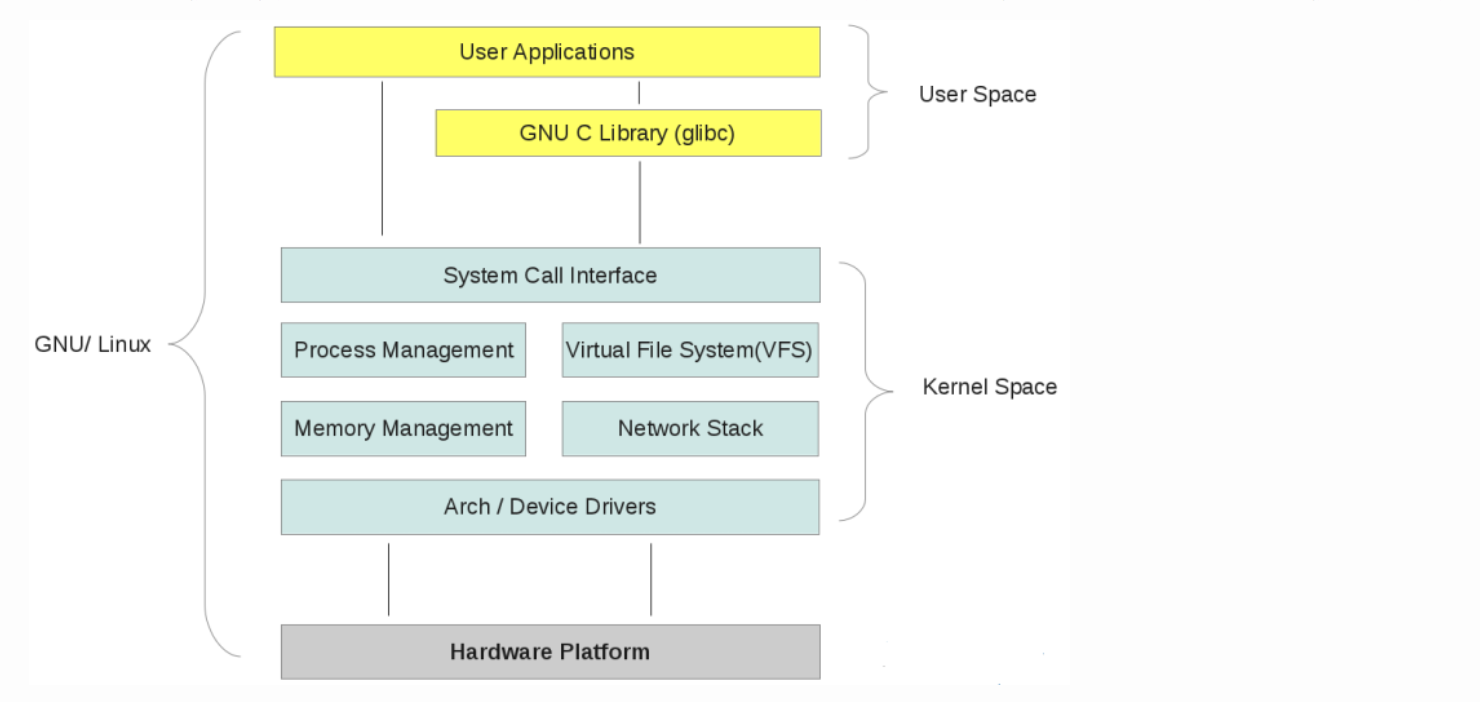
应用程序想要访问内核资源，怎么办，有三种方法：系统调用、异常(中断)和陷入。

应用程序不会直接调用系统调用，而是通过API函数来间接的调用系统调用，比如POSIX、API和C库等。unix类操作系统中最常用的编程接口就是POSIX。

应用 程序使用open函数 打开一个设备文件。

每个系统调用都有一个系统调用号。

系统调用处于内核空间，应用程序无法直接访问，因此需要“陷入“到内核，方法就是软中断。陷入内核以后还要指定系统调用号。



# 二、字符设备驱动开发流程

1，Linux里面一切皆文件，驱动设备表现就是一个/dev/下的文件，/dev/led。应用程序调用open函数打开设备，比如led。应用程序通过write函数向/dev/led写数据，比如写1表示打开，写0表示关闭。如果要关闭设备那么就是close函数。

2，编写驱动的 时候也需要编写驱动对应的open、close，write函数。字符设备驱动fileoptions\_struct.

1、驱动最终是被应用调用的，在写驱动的时候要考虑应用开发的便利性。

2、驱动是分驱动框架的，要按照驱动框架来编写，对于字符设备驱动来说，重点编写应用程序对应的open、close、read、write等函数。