一、起源

仅devfs，导致开发不方便以及一些功能难以支持：

1. 热插拔
2. 不支持一些针对所有设备的统一操作（如电源管理）
3. 不能自动mknod
4. 用户查看不了设备信息
5. 设备信息硬编码，导致驱动代码通用性差，即没有分离设备和驱动

二、新方案

uevent机制：sysfs + uevent + udevd（上层app）

2.1 sysfs: 一种用内存模拟的文件系统，系统启动时mount到/sys目录

sysfs用途：（类似于windows的设备管理器）

1. 建立系统中总线、驱动、设备三者之间的桥梁
2. 向用户空间展示内核中各种设备的拓扑图
3. 提供给用户空间对设备获取信息和操作的接口，部分取代ioctl功能

sysfs在内核中的组成要素	在用户空间/sys下的显示
内核对象（kobject）	目录
对象属性（attribute）	文件
对象关系（relationship）	链接（Symbolic Link）

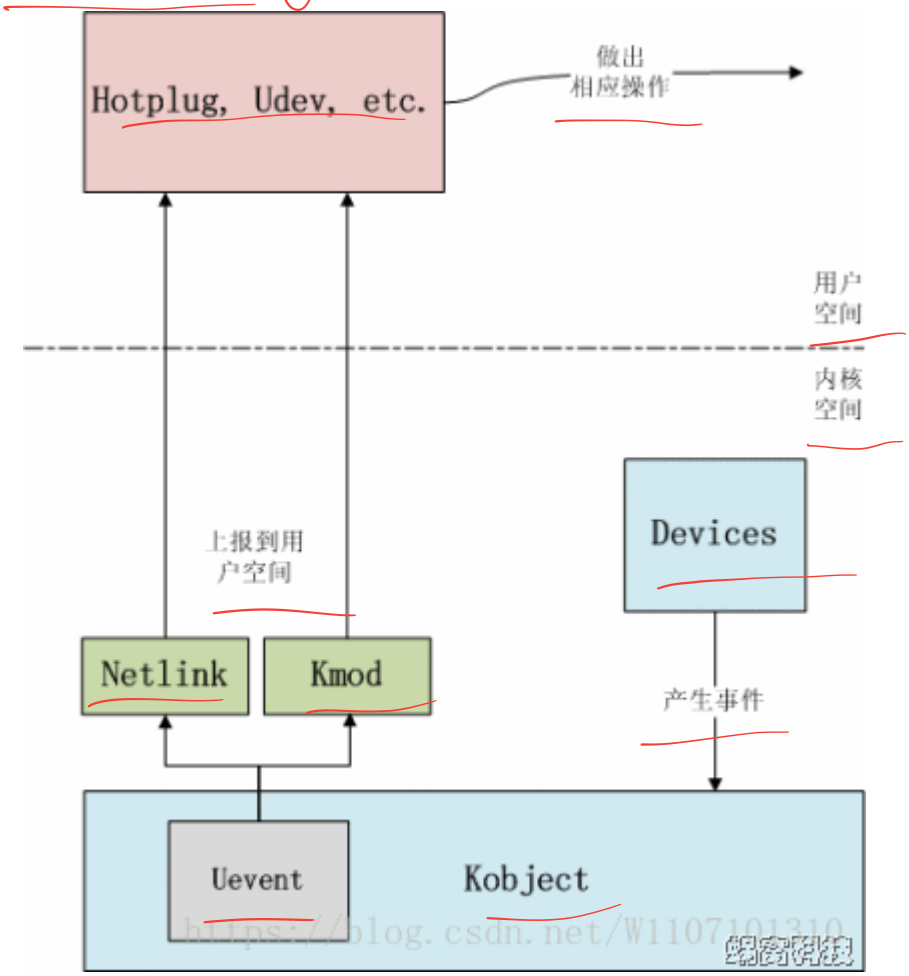
四个基本结构

类型	所包含的内容	内核数据结构	对应/sys项
设备(Devices)	设备是此模型中最基本的类型，以设备本身的连接按层次组织	struct device	/sys/devices/?/?/.../
驱动(Drivers)	在一个系统中安装多个相同设备，只需要一份驱动程序的支持	struct device_driver	/sys/bus/pci/drivers/?/
总线(Bus)	在整个总线级别对此总线上连接的所有设备进行管理	struct bus_type	/sys/bus/?/
类别(Classes)	这是按照功能进行分类组织的设备层次树；如 USB 接口和 PS/2 接口的鼠标都是输入设备，都会出现在/sys/class/input/下	struct class	/sys/class/?/

目录组织结构：

/sys下的子目录	所包含的内容
/sys/devices	这是内核对系统中所有设备的分层次表达模型，也是/sys文件系统管理设备的最重要的目录结构；
/sys/dev	这个目录下维护一个按字符设备和块设备的主次号码(major:minor)链接到真实的设备(/sys/devices下)的符号链接文件；
/sys/bus	这是内核设备按总线类型分层放置的目录结构， devices 中的所有设备都是连接于某种总线之下，在这里的每一种具体总线之下可以找到每一个具体设备的符号链接，它也是构成 Linux 统一设备模型的一部分；
/sys/class	这是按照设备功能分类的设备模型，如系统所有输入设备都会出现在/sys/class/input 之下，而不论它们是以何种总线连接到系统。它也是构成 Linux 统一设备模型的一部分；
/sys/kernel	这里是内核所有可调整参数的位置，目前只有 uevent_helper, kexec_loaded, mm, 和新式的 slab 分配器等几项较新的设计在使用它，其它内核可调整参数仍然位于 sysctl(/proc/sys/kernel) 接口中；
/sys/module	这里有系统中所有模块的信息，不论这些模块是以内联(inlined)方式编译到内核映像文件(vmlinuz)中还是编译为外部模块(ko文件)，都可能会出现在/sys/module 中
/sys/power	这里是系统中电源选项，这个目录下有几个属性文件可以用于控制整个机器的电源状态，如可以向其中写入控制命令让机器关机、重启等。

2.2 uevent



三、代码中自动mknod

```
struct class *class_create(struct module *owner, const char *name);
/*
 * 功能：在/sys/class生成一个目录,目录名由name指定
 * 参数：
    struct module *owner - THIS_MODULE
    const char *name - 目录名
 * 返回值  成功：class指针   失败：NULL
*/
/*
辅助接口：可以定义一个struct class 的指针变量cls来接受返回值，然后通过IS_ERR(cls)判断是否失败；
IS_ERR(cls);成功----------------->0
IS_ERR(cls);失败----------------->非0
PTR_ERR(cls);来获得失败的返回错误码；
*/
```

```
void class_destroy(struct class *cls)
/*
* 功能：删除class_create生成目录
* 参数：
    struct class *cls - class指针
* 返回值
*/
```

```
struct device *device_create(struct class *class, struct device *parent,
                dev_t devt, void *drvdata, const char *fmt, ...)
/*
 * 功能：在/sys/class目录下class_create生成目录再生成一个子目录与该设备相对应，发uevent让应用程序udevd创建设备文件
 * 参数：
    struct class *class - class指针
    struct device *parent - 父对象，一般NULL
    dev_t devt - 设备号
    void *drvdata - 驱动私有数据，一般NULL
    const char *fmt - 字符串的格式
     ... - 不定参数
 * 返回值
    成功：device指针
    失败：NULL
 */
```

× init 创建两
exit 清这两就好

```
void device_destroy(struct class *class, dev_t devt)
/*
 * 功能：删除device_create生成目录
 * 参数：
    struct class *class - class指针
    dev_t devt - 设备号
 * 返回值
*/
```