**Class概念的说明**

* 类是一个设备的高层视图，抽象出了设备的共性。
* 类允许用户空间使用设备所提供的功能。
* 几乎所有的类都显示在/sys/class目录下，如网络接口在/sys/class/net/目录下
* 多数情况下，类子系统是向用户空间导出信息的最好办法。
* 驱动核心管理类的两个不同类型的接口：class\_simple和class
* 在插拔设备的时候，类会产生热插拔事件。

**class\_simple接口**

* **作用说明：**
  + 向系统中添加新的类
  + 一般这些接口的主要目的是，提供包含设备号的**属性文件，**以便创建设备节点。
* **创建类的接口**
  + struct class\_simple \*class\_simple\_create(struct module \*owner, char \*name);
    - name: 创建的类名
* **销毁一个类的接口**
  + void class\_simple\_destroy(struct class\_simple \*cs);
* **向类中添加设备的接口**
  + struct class\_device \*class\_simple\_device\_add(struct class\_simple \*cs, dev\_t dev, struct device \*device, const char \*fmt, ...);
    - dev:设备号
    - device:非NULL的时候，一个符号链接将指向/sys/devices/下的设备入口
  + 可以调用class\_device\_create\_file()接口，来为设备创建属性文件。
* **热插拔事件产生时，为用户空间处理程序添加环境变量的回调函数注册接口是**
  + int class\_simple\_set\_hotplug(struct class\_simple \*cs,

int (\*hotplug)(struct class\_device \*dev, char \*\*envp, int num\_envp, char \*buffer, int buffer\_size));

* **删除设备的入口是**：
  + void class\_simple\_device\_remove(dev\_t dev)

**完整的类接口**

* **描述类的结构**

struct class {

char \* name; //在/sys/class/目录下显示的类的名称

struct subsystem subsys;

struct list\_head children;

struct list\_head interfaces;

struct class\_attribute \* class\_attrs; //当类被注册的时候，其指向的所有设备属性都会被创建

struct class\_device\_attribute \* class\_dev\_attrs; //设备的默认属性

int (\*hotplug)(struct class\_device \*dev, char \*\*envp, int num\_envp, char \*buffer, int buffer\_size); //热插拔事件发生时，回调为用户空间程序添加环境变量

void (\*release)(struct class\_device \*dev); //将设备从类中删除时的回调函数

void (\*class\_release)(struct class \*class); //类被删除时的回调函数

};

* **描述类属性的结构**

struct class\_attribute {

struct attribute attr;

ssize\_t (\*show)(struct class \*, char \* buf); //读取属性的回调接口

ssize\_t (\*store)(struct class \*, const char \* buf, size\_t count); //写属性的回调借接口

};

* **初始化类属性的宏**
  + CLASS\_ATTR(\_name,\_mode,\_show,\_store)
* **创建类属性文件的接口**
  + int device\_create\_file(struct device \*device, struct device\_attribute \* entry);
* **移除类属性文件的接口**
  + void device\_remove\_file(struct device \* dev, struct device\_attribute \* attr);
* **注册类的接口**
  + **int class\_register(struct class \*);**
* **注销类的接口**
  + **void class\_unregister(struct class \*);**
* **类存在的目的是**
  + 给类成员的各个设备充当一个容器。
* **描述类成员的结构体**

struct class\_device {

struct list\_head node;

struct kobject kobj;

struct class \* class; /\* required \*/ //指向包含设备的类

struct device \* dev; /\* not necessary, but nice to have \*/ //可选，非NULL，将是指向/sys/devices/下的设备符号链接

void \* class\_data; /\* class-specific data \*/ //保存私有数据

char class\_id[BUS\_ID\_SIZE]; /\* unique to this class \*/ //在sysfs中显示的设备名

};

* **注册类设备的接口**
  + int class\_device\_register(struct class\_device \*);
* **注销类设备的接口**
  + void class\_device\_unregister(struct class\_device \*);
* **更改类设备名字的接口**
  + int class\_device\_rename(struct class\_device \*, char \*);
* **描述类设备属性的结构**

struct class\_device\_attribute {

struct attribute attr;

ssize\_t (\*show)(struct class\_device \*, char \* buf);

ssize\_t (\*store)(struct class\_device \*, const char \* buf, size\_t count);

};

* **初始化类设备属性的宏**
  + #define CLASS\_DEVICE\_ATTR(\_name,\_mode,\_show,\_store)
* **创建类设备属性文件的接口**
  + int class\_device\_create\_file(struct class\_device \*, const struct class\_device\_attribute \*);
* **删除类设备属性文件的接口**
  + void class\_device\_remove\_file(struct class\_device \*, const struct class\_device\_attribute \*);

**类接口**

* **向类中添加设备或者从类中删除设备时的触发机制**
  + 说明：
    - 当向类中添加设备时会调用回调函数add的执行，并触发一些信息
    - 当从类中删除设备时会调用remove函数的执行，并触发一些信息
* **描述接口的结构**

struct class\_interface {

struct list\_head node;

struct class \*class;

int (\*add) (struct class\_device \*);

void (\*remove) (struct class\_device \*);

};

* **注册接口的函数**
  + int class\_interface\_register(struct class\_interface \*);
* **注销接口的函数**
  + void class\_interface\_unregister(struct class\_interface \*);