Device Classes

Introduction

A device class describes a type of device, like an audio or network device. The following device classes have been identified:

<Insert List of Device Classes Here>

Each device class defines a set of semantics and a programming interface that devices of that class adhere to. Device drivers are the implementation of that programming interface for a particular device on a particular bus.

Device classes are agnostic with respect to what bus a device resides on.

Programming Interface

The device class structure looks like:

```
typedef int (*devclass_add)(struct device *);
typedef void (*devclass_remove)(struct device *);

struct device_class {
        char                    * name;
        rwlock_t                lock;
        u32                     devnum;
        struct list_head        node;

        struct list_head        drivers;
        struct list_head        intf_list;

        struct driver_dir_entry dir;
        struct driver_dir_entry device_dir;
        struct driver_dir_entry driver_dir;

        devclass_add            add_device;
        devclass_remove         remove_device;
};
```

A typical device class definition would look like:

```
struct device_class input_devclass = {
        .name           = "input",
        .add_device     = input_add_device,
        .remove_device  = input_remove_device,
};
```

Each device class structure should be exported in a header file so it can be used by drivers, extensions and interfaces.

Device classes are registered and unregistered with the core using:

```
int devclass_register(struct device_class * cls);
void devclass_unregister(struct device_class * cls);
```

Devices

As devices are bound to drivers, they are added to the device class that the driver belongs to. Before the driver model core, this would typically happen during the driver's probe() callback, once the device has been initialized. It now happens after the probe() callback finishes from the core.

The device is enumerated in the class. Each time a device is added to the class, the class's devnum field is incremented and assigned to the device. The field is never decremented, so if the device is removed from the class and re-added, it will receive a different enumerated value.

The class is allowed to create a class-specific structure for the device and store it in the device's class_data pointer.

There is no list of devices in the device class. Each driver has a list of devices that it supports. The device class has a list of drivers of that particular class. To access all of the devices in the class, iterate over the device lists of each driver in the class.

Device Drivers

Device drivers are added to device classes when they are registered with the core. A driver specifies the class it belongs to by setting the struct device_driver::devclass field.

sysfs directory structure

There is a top-level sysfs directory named 'class'.

Each class gets a directory in the class directory, along with two default subdirectories:

```
        class/
        `-- input
            |-- devices
            `-- drivers
```

Drivers registered with the class get a symlink in the drivers/ directory that points to the driver's directory (under its bus directory):

```
   class/
   `-- input
       |-- devices
```

```
    `-- drivers
        `-- usb:usb_mouse -> ../../../bus/drivers/usb_mouse/
```

Each device gets a symlink in the devices/ directory that points to the device's directory in the physical hierarchy:

```
   class/
   `-- input
       |-- devices
       |   `-- 1 -> ../../../root/pci0/00:1f.0/usb_bus/00:1f.2-1:0/
       `-- drivers
```

Exporting Attributes

```
struct devclass_attribute {
        struct attribute        attr;
        ssize_t (*show)(struct device_class *, char * buf, size_t count, loff_t
off);
        ssize_t (*store)(struct device_class *, const char * buf, size_t count,
loff_t off);
};
```

Class drivers can export attributes using the DEVCLASS_ATTR macro that works similarly to the DEVICE_ATTR macro for devices. For example, a definition like this:

```
static DEVCLASS_ATTR(debug,0644,show_debug,store_debug);
```

is equivalent to declaring:

```
static devclass_attribute devclass_attr_debug;
```

The bus driver can add and remove the attribute from the class's sysfs directory using:

```
int devclass_create_file(struct device_class *, struct devclass_attribute *);
void devclass_remove_file(struct device_class *, struct devclass_attribute *);
```

In the example above, the file will be named 'debug' in placed in the class's directory in sysfs.

Interfaces

There may exist multiple mechanisms for accessing the same device of a particular class type. Device interfaces describe these mechanisms.

When a device is added to a device class, the core attempts to add it to every interface that is registered with the device class.