Linux Device Driver
Character Device Driver-CDD

https://community.ruggedboard.com

Character Device Driver-CDD

Rugged Board

Kernel APIs and utilities to be used in driver code

```
alloc_chrdev_region();
```

1.Create device number

```
cdev_init();
cdev_add();
```

2. Make a char device registration with the VFS

```
class_create();
device_create();
```

3. Create device files

Character Device Driver-CDD

Creating Device File:

Device file can be created in two ways:

1. Manual
2. Automatic

Manual:

We can create the device file manually by using mknod.

$ mknod -m <permissions> <name> <device type> <major> <minor>

-m <permissions> – optional argument that sets the permission bits of the new device file to permissions

<name> – your device file name that should have full path (/dev/name)

<device type> – Put c or b c – Character Device b – Block Device

<major> – major number of your device

<minor> – minor number of your driver

Eg. $sudo mknod -m 0644 /dev/mydevice c 244 10

Character Device Driver-CDD

Automatic:

Traditionally, device nodes were stored in the /dev directory on Linux systems.

There was a node for every possible type of device, regardless of whether it actually existed in the system.

The result was that this directory took up a lot of space.

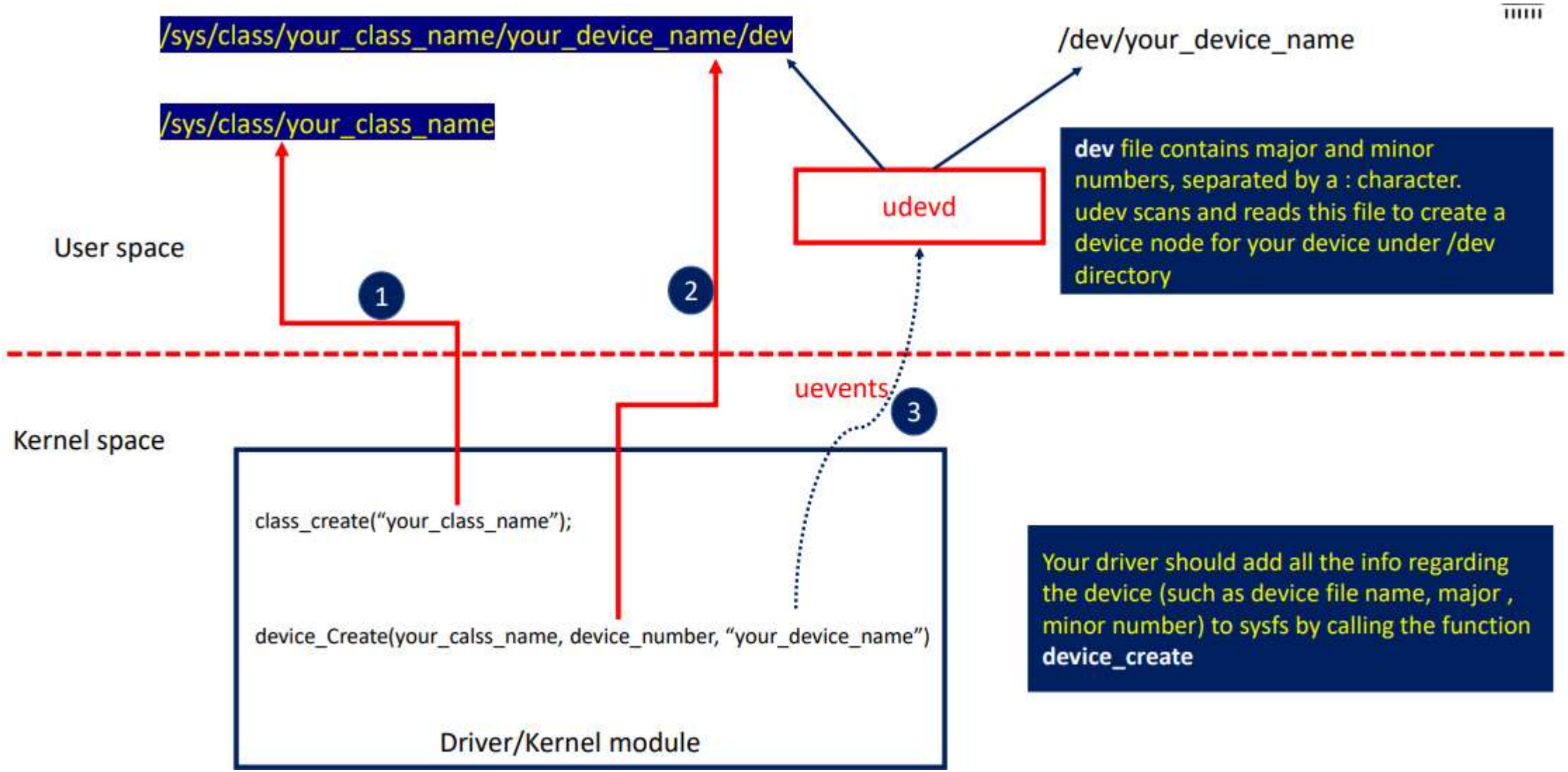
udev introduces a new way of creating device nodes.

It compares the information made available by sysfs and creates nodes.

udev can be further configured using its configuration files to tune the device file names, their permissions, their types, etc.

So, as far as driver is concerned, the appropriate /sys entries need to be populated using the Linux device model APIs declared in <linux/device.h> and the rest would be handled by udev.

Character Device Driver-CDD



Character Device Driver-CDD

class_create — create a struct class structure

struct class * class_create (struct module *owner,
const char *name);

owner - pointer to the module that is to “own” this struct class

name - pointer to a string for the name of this class.

Header File: <linux/device.h>

Now, the name will appear in /sys/class/<name>.

Description:

This is used to create a struct class pointer that can then be used in calls to class_device_create.

class_destroy — destroys a struct class structure

void class_destroy (struct class *cls);

Character Device Driver-CDD

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/kdev_t.h>
#include <linux/fs.h>
#include <linux/device.h>

int base_minor = 0;
char *device_name = "mychardev";
int count = 1;
dev_t devicenumber;

static struct class *class;
static struct device *device;

MODULE_LICENSE("GPL");
static int test_hello_init(void)
{
  class = class_create(THIS_MODULE, "myclass");
if (!alloc_chrdev_region(&devicenumber, base_minor, count, device_name))
{
      device = device_create(class, NULL, devicenumber, NULL, device_name);

}
else
       printk("Device number registration Failed\n");

     return 0;
}

static void test_hello_exit(void)
{
            unregister_chrdev_region(devicenumber, count);
            device_destroy(class, devicenumber);
            class_destroy(class);
}

module_init(test_hello_init);
module_exit(test_hello_exit);
```

Character Device Driver-CDD

$ udevadm monitor

With this command, we can tap into udev in real time and see what it sees when we plug in different devices

Connect-us

Developer Wiki

Open Discussions