Linux Device Driver
Basic Character Device Driver

https://community.ruggedboard.com

Basic Character Device Driver

Character Device Driver

A character device typically transfers data to and from a user application

- — they behave like pipes or serial ports
- — instantly reading or writing the byte data in a character-by-character stream.

Steps in creating a character driver

1. Allocate a device number dynamically or statically (dev_t)
2. Initializing the character device with its file operations (struct cdev, struct file_operations)
3. Registering the character device with Linux Kernel (cdev_add)

We have three things

1. Application
2. Device File
3. Device Driver

How user application communicate with device driver?

Basic Character Device Driver

Application ----- Device File

Connection between application and Device File is based on Device Name

Device Driver------Device File

Connection between Device Driver and Device File is based on Device Number/Device ID

A device ID/number consists of two parts:

Major Number : identifies the device type (IDE disk, SCSI disk, serial port, etc.)

Minor Number : identifies the device (first disk, second serial port, etc.)

Most times, the major number identifies the driver, while the minor number identifies each physical device served by the driver.

Basic Character Device Driver

Certain major identifiers are statically assigned to devices
(in the Documentation/admin-guide/devices.txt file from the kernel sources).

When choosing the identifier for a new device, you can use two methods:

- static (choose a number that does not seem to be used already)
- dynamic (kernel will give you a device number)

Data Type:

- A device ID/number is represented using the type dev_t.
- 12 bit major number + 20 bit Minor number =32 bit dev_t

To obtain the major or minor parts of a dev_t:

- MAJOR(dev_t dev);
- MINOR(dev_t dev);

To create a device number from major and minor number:

- MKDEV(int major, int minor);
- Header File: linux/kdev_t.h

Basic Character Device Driver

/proc/devices

- This file displays the various character and block devices currently configured
- The output from /proc/devices includes the major number and name of the device

Output of this file is broken into two major sections:

- Character devices
- Block devices.

Basic Character Device Driver

Allocating Major and Minor NumberTwo ways:

1. Static
2. Dynamic

Difference between static and dynamic method

- Static method is only really useful if you know in advance which major number you want to start with.
- With Static method , you tell the kernel what device numbers you want (the start major/minor number and count) and it either gives them to you or not (depending on availability).
- With Dynamic method, you tell the kernel how many device numbers you need (the starting minor number and count) and it will find a starting major number for you, if one is available, of course.
- Partially to avoid conflict with other device drivers, it’s considered preferable to use the Dynamic method function, which will dynamically allocate the device numbers for you.

Basic Character Device Driver

Static assignment and unallocation of device identifiers:

int register_chrdev_region (dev_t from,unsigned count,const char *name);

Description:

Register a range of device numbers

Arguments:

from : the first in the desired range of device numbers; must include the major number.

count: the number of consecutive device numbers required

name: the name of the device or driver.

This will appear in /proc/devices

Return Value: zero on success, a negative error code on failure.

In module_exit use the following function to unregister the driver:

void unregister_chrdev_region(dev_t from, unsigned int count);

Header File: <linux/fs.h>

Basic Character Device Driver

mychar_driver.c

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

int major_number = 120;
int minor_number = 0;
char *device_name = "mychardev";
int count = 1;
dev_t devicenumber;

module_param(major_number, int, 0);
module_param(minor_number, int, 0);
module_param(count, int, 0);
module_param(device_name, charp, 0);

MODULE_LICENSE("GPL");
```

```
static int test_hello_init(void)
{
        devicenumber = MKDEV(major_number, minor_number);
        printk("Major Number :%d\n", MAJOR(devicenumber));
        printk("Minor Number :%d\n", MINOR(devicenumber));
        printk("Count:%d\n", count);
        printk("Device Name:%s\n", device_name);

        if (!register_chrdev_region(devicenumber, count, device_name))
                printk("Device number registered\n");
        else
                printk("Device number registration Failed\n");

        return 0;
}
static void test_hello_exit(void)
{
        unregister_chrdev_region(devicenumber, count);
}

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

Basic Character Device Driver

Try these commands :

- sudo insmod ./hello.ko major_number=126 && cat /proc/devices | less
- sudo insmod ./hello.ko major_number=128
- sudo insmod ./hello.ko major_number=126 device_name=RB
- sudo insmod ./hello.ko major_number=126 minor_number=0 count=1048576 device_name=RB
- sudo insmod ./hello.ko major_number=126 minor_number=10 count=1048576 device_name=RB

NOTE:

- Maximum minor number allowed is 1048576 (2^20 20 bits for minor number)
- Maximum major number allowed is CHRDEV_MAJOR_MAX as an artificial limit (chosen to be 511 even though 12 bits are used) file: linux/fs.h

Basic Character Device Driver

Dynamic Allocation:

If we don't want fixed major and minor number use this method.

This method will allocate the major number dynamically to your driver which is available.

```
int alloc_chrdev_region (dev_t *  dev,
                         unsigned          baseminor,
                  unsigned      count,
                          const char * name);
```

Description:

Allocates a range of char device numbers.

The major number will be chosen dynamically, and returned (along with the first minor number) in dev

Arguments:

dev	-->	output parameter for first assigned number
baseminor	-->	first of the requested range of minor numbers
count	-->	the number of minor numbers required
name	-->	the name of the associated device or driver

Return Value:

Returns zero or a negative error code.

Basic Character Device Driver

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

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

module_param(base_minor, int, 0);
module_param(count, int, 0);
module_param(device_name, charp, 0);

MODULE_LICENSE("GPL");
static int test_hello_init(void)
{

        printk("Minor Number :%d\n", base_minor);
        printk("Count:%d\n", count);
        printk("Device Name:%s\n", device_name);
if (!alloc_chrdev_region(&devicenumber, base_minor, count,
device_name)) {
                printk("Device number registered\n");
                printk("Major number received:%d\n",
MAJOR(devicenumber));
        }
        else
                printk("Device number registration
Failed\n");

        return 0;
}

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

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

Basic Character Device Driver

```
obj-m += hello.o

all:
	make -C /lib/modules/`uname -r`/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Commands:

```
sudo insmod ./hello.ko
sudo insmod ./hello.ko base_minor=1048576 count=10 device_name=usb
```

Connect-us

Developer Wiki

Open Discussions