



Sidebar▼

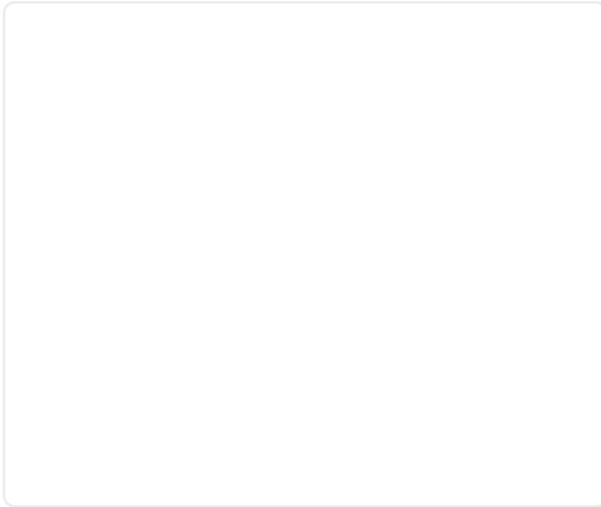
[Home](#) → [Tutorials](#) → [Linux](#) → [Device Drivers](#) → **Linux Device Driver**
Tutorial Part 3 - Module Parameter

Device Drivers



Linux Device Driver Tutorial Part 3 - Module Parameter

8



This is the [Series on Linux Device Driver](#). The aim of this series is to provide the easy and practical examples of Linux Device Drivers that anyone can understand easily. Now we are going to see Linux Device Driver Tutorial Part 3 – Passing Arguments to Device Driver.

Apple Music Turns 5 as It Continues Rivalry With Spotify

8

Post Contents [\[hide\]](#)

- 1 [Linux Device Driver Tutorial Part 3 – Passing Arguments to Device Driver](#)
- 2 [Module Parameters Macros](#)
 - 2.1 [module_param\(\)](#)
 - 2.2 [module_param_array\(\)](#)

2.3 module_param_cb()

2.3.1 When we will need this notification?

3 Programming

4 Compiling

5 Loading the Driver

6 Verify the parameters by using dmesg

7 Unloading the Driver

7.0.1 Share this:

7.0.2 Like this:

7.0.3 Related

Linux Device Driver Tutorial Part 3 - Passing Arguments to Device Driver

We can pass the arguments to any other functions in same program. But Is it possible to pass any arguments to any program? I think Probably yes. Right? Well, we can. In C Programming we can pass the arguments to the program. For that we need to add **argc** and **argv** in main function definition. I hope everyone knows that. Now come to our topic. Is it possible to pass the argument to the Device Driver? Fine. In this tutorial we are going to see that topic.

8

Module Parameters Macros

- **module_param()**
- **module_param_array()**
- **module_param_cb()**

Before discuss these macros we have to know about permissions of the variable.

There are several types of permissions:

- S_IWUSR
- S_IRUSR
- S_IXUSR
- S_IRGRP
- S_IWGRP
- S_IXGRP

In this S_I is common header.

R = read ,W =write ,X= Execute.

USR =user ,GRP =Group

Using OR '|' (or operation) we can set multiple permissions at a time.

module_param()

This macro used to initialize the arguments. **module_param** takes three parameters: the name of the variable, its type, and a permissions mask to be used for an accompanying sysfs entry. The macro should be placed outside of any function and is typically found near the head of the source file. **module_param()** macro, defined in **linux/moduleparam.h**.

```
module_param(name, type, perm);
```

module_param() macro creates the sub-directory under **/sys/module**. For example

```
1 module_param(valueETX, int, S_IWUSR|S_IRUSR);
```

This will create the sysfs entry. (**/sys/module/hello_world_module/parameters/valueETX**)

Numerous types are supported for module parameters:

- bool
- invbool

A boolean (true or false) value (the associated variable should be of type int). The **invbool** type inverts the value, so that true values become false and vice versa.

- charp

A char pointer value. Memory is allocated for user-provided strings, and the pointer is set accordingly.

- int
- long
- short
- uint
- ulong
- ushort

Basic integer values of various lengths. The versions starting with u are for unsigned values.

module_param_array()

This macro is used to send the array as a argument. Array parameters, where the values are supplied as a comma-separated list, are also supported by the module loader. To declare an array parameter, use:

```
module_param_array(name, type, num, perm) ;
```

Where,

name is the name of your array (and of the parameter),

type is the type of the array elements,

num is an integer variable (optional) otherwise NULL, and

8

perm is the usual permissions value.

module_param_cb()

This macro used to register the callback whenever the argument (parameter) got changed. I think you don't understand. Let me explain

properly.

For Example,

I have created one parameter by using `module_param()`.

```
1 module_param(valueETX, int, S_IWUSR|S_IRUSR);
```

This will create the sysfs entry. (`/sys/module/hello_world_module/parameters/valueETX`)

You can change the value of `valueETX` from the command line by

```
1 sudo su
2 echo 1 > /sys/module/hello_world_module/parameters/valueETX
```

This will update the `valueETX` variable. But there is no way to notify your module that “`valueETX`” has changed.

By using this `module_param_cb()` macro, we can get a notification.

If you want to get notification whenever value got change. we need to register our handler function to its file operation structure.

```
1 struct kernel_param_ops {
2     int (*set)(const char *val, const struct kernel_param *kp);
3     int (*get)(char *buffer, const struct kernel_param *kp);
4     void (*free)(void *arg);
5 };
```

For further explanation, please refer below program.

8 When we will need this notification?

I will tell you about the practical scenario. Whenever value is set to 1, you have to write something into a hardware register. How can you do this if the change of value variable is not notified to you? Got it? I think you have understood. If you

didn't understand, just see the explanation posted below.

Programming

In this example, i explained all (module_param, module_param_array, module_param_cb).

For module_param(), I have created two variables. One is integer (**valueETX**) and another one is a string (**nameETX**).

For module_param_array(), i have created one integer array variable (**arr_valueETX**).

For module_param_cb(), i have created one integer variable (**cb_valueETX**).

You can change the all variable using their sysfs entry which is under **/sys/module/hello_world_module/parameters/**

But you won't get any notification when they got change, except the variable which is created by **module_param_cb()** macro.

Download the code by clicking below link.

8

[\[Download Project Here\]](#)

```
1 #include<linux/kernel.h>
2 #include<linux/init.h>
3 #include<linux/module.h>
```

```

4  #include<linux/moduleparam.h>
5
6  int valueETX, arr_valueETX[4];
7  char *nameETX;
8  int cb_valueETX = 0;
9
10 module_param(valueETX, int, S_IRUSR|S_IWUSR);           //integer value
11 module_param(nameETX, charp, S_IRUSR|S_IWUSR);           //String
12 module_param_array(arr_valueETX, int, NULL, S_IRUSR|S_IWUSR); //Array of integers
13
14 /*-----Module_param_cb()-----*/
15 int notify_param(const char *val, const struct kernel_param *kp)
16 {
17     int res = param_set_int(val, kp); // Use helper for write variable
18     if(res==0) {
19         printk(KERN_INFO "Call back function called...\n");
20         printk(KERN_INFO "New value of cb_valueETX = %d\n", cb_valueETX);
21         return 0;
22     }
23     return -1;
24 }
25
26 const struct kernel_param_ops my_param_ops =
27 {
28     .set = &notify_param, // Use our setter ...
29     .get = &param_get_int, // .. and standard getter
30 };
31
32 module_param_cb(cb_valueETX, &my_param_ops, &cb_valueETX, S_IRUGO|S_IWUSR );
33 /*-----*/
34
35 static int __init hello_world_init(void)
36 {
37     int i;
38     printk(KERN_INFO "ValueETX = %d \n", valueETX);
39     printk(KERN_INFO "cb_valueETX = %d \n", cb_valueETX);
40     printk(KERN_INFO "NameETX = %s \n", nameETX);
41     for (i = 0; i < (sizeof arr_valueETX / sizeof (int)); i++) {
42         printk(KERN_INFO "Arr_value[%d] = %d\n", i, arr_valueETX[i]);
43     }
44     printk(KERN_INFO "Kernel Module Inserted Successfully...\n");
45     return 0;
46 }
47
48 void __exit hello_world_exit(void)
49 {
50     printk(KERN_INFO "Kernel Module Removed Successfully...\n");
51 }
52
53 module_init(hello_world_init);
54 module_exit(hello_world_exit);
55
56 MODULE_LICENSE("GPL");
57 MODULE_AUTHOR("EmbeTronicX <embetronicx@gmail.com or admin@embetronicx.com>");
58 MODULE_DESCRIPTION("A simple hello world driver");
59 MODULE_VERSION("1.0");

```

Compiling

This is the code of **Makefile**.


```
1 obj-m += hello_world_module.o
2
3 KDIR = /lib/modules/${shell uname -r}/build
4
5 all:
6     make -C $(KDIR) M=$(shell pwd) modules
7
8 clean:
9     make -C $(KDIR) M=$(shell pwd) clean
```

In terminal enter **sudo make**

Loading the Driver

```
sudo insmod hello_world_module.ko valueETX=14
nameETX="EmbeTronicX" arr_valueETX=100,102,104,106
```

Verify the parameters by using dmesg

Now our module got loaded. now check **dmesg**. In the below picture, every value got passed to our device driver.

8

Now I'm going to check **module_param_cb()** is whether calling that handler function or not. For that, I need to change the variable in sysfs. You can write that variable by two ways.

1. **sudo sh -c "echo 13 > /sys/module/driver/parameters/cb_valueETX"**
2. Type **sudo su**. Then enter the password if it asks. Then do **echo 13 > /sys/module/hello_world_module/parameters/cb_valueETX**

Now do **dmesg** and check.

See the above result. So Our callback function got called. But if you change the value of other variables, you won't get the notification.

Unloading the Driver

Finally unload the driver by using **sudo rmmod hello_world_module**.

I hope you understood. If you have any doubt, please comment below. 😊

5

Article Rating



8

Share this:

Share 0

WhatsApp

Share

Telegram

More

Tweet

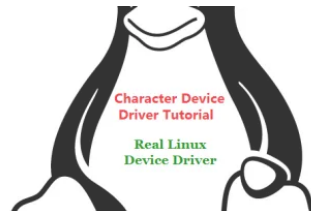
Print

Email

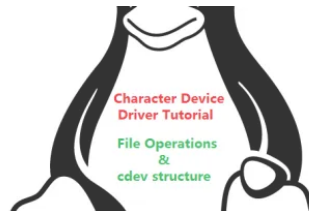
Like this:

Loading...

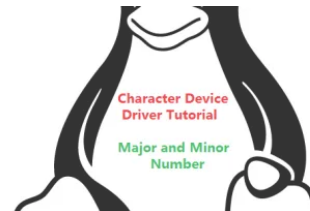
Related



Linux Device Driver
Tutorial Part 7 - Linux
Device Driver Tutorial
Programming
In "Device Drivers"



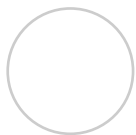
Linux Device Driver
Tutorial Part 6 - Cdev
structure and File
Operations
In "Device Drivers"



Linux Device Driver
Tutorial Part 4 -
Character Device Driver
In "Device Drivers"

☒ Subscribe ▼

Connect with | [Login](#)



Join the discussion

B *I* U **[+]**



This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

8 COMMENTS



Oldest ▼



Ajinkya

May 22, 2018 1:29 PM

8

Thanks a ton, guys for such a great series of articles. I've never seen such informative and detailed contents. Thanks again!

Loading...



0



Reply



himanshu

August 15, 2018 10:29 PM

Awesome tutorial for device driver in linux love the way you are explaining each totorial.

Can you please upload a tutorial for LED ON/OFF in Raspberry pi without using predefined API for RPi.

Loading...



0



Reply



Mohan

November 29, 2018 12:06 AM

Hi, I think `.set = ¬ify_param, // Use our setterget = ¶m_get_int, // ..` and standard getter here `&` operator is not required as it is a function and to pass a function simply we have to write the name of function. So we have to use below mentioned thing which won't through any warning. `.set = notify_param, // Use our setterget = param_get_int, // ..` and standard getter Also please check, i think this won't work fine as after giving this below mentioned command `echo 13 > /sys/module/hello_world_module/parameters/cb_valueETX` we are getting `bash: echo: write error:...` [Read more »](#)



0




Reply

8



EmbeTronicX



 Reply to [Mohan](#)

April 1, 2020 4:46 AM

Hi Mohan,

For functions, we can put `&func_name` or just `func_name` is enough to take the function's address. We can use any one of the method. And for the write permission, we have updated the way of written the variable. Please check it out.

Loading...

 0   Reply



Dheeraj


September 10, 2019 6:37 AM

What should be the operation if I want to update the value of valueETX after insmod command ?

Loading...

 0   Reply

EmbeTronicX

 Reply to [Dheeraj](#)

April 1, 2020 4:36 AM



Hi Dheeraj,

You can write through the sysfs command. Please use the below command to write.

```
sudo sh -c "echo 18 > /sys/module/driver  
/parameters/valueETX"
```

But you won't get notification like the variable `cb_valueETX`. If you want to get notification callback, then you have to use `module_param_cb`.

Loading...

 0   Reply

8



Bernd W.

January 14, 2020 2:57 PM

Hi,

I think there is an error in line 32:

```
module_param_cb(cb_valueETX, &my_param_ops,  
&cb_valueETX, S_IRUGO|S_IWUSR );  
^^^
```

I had to set the write-permission indicated above to "IWUGO", to make the value update via "echo 13 > /sys/[...]/cb_valueETX" work. Before that change, I always got a permission error - even with "sudo".

Loading...



0



Reply

EmbeTronicX

Reply to [Bernd W.](#)

April 1, 2020 4:36 AM

Hi Brend,

when you use S_WUGO, it will throw an error while compiling. Having module parameter's writable by non-privileged user normally is not good for security reasons. So we have updated the other way of writing the variable. Please check it.

Loading...

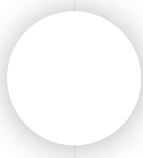


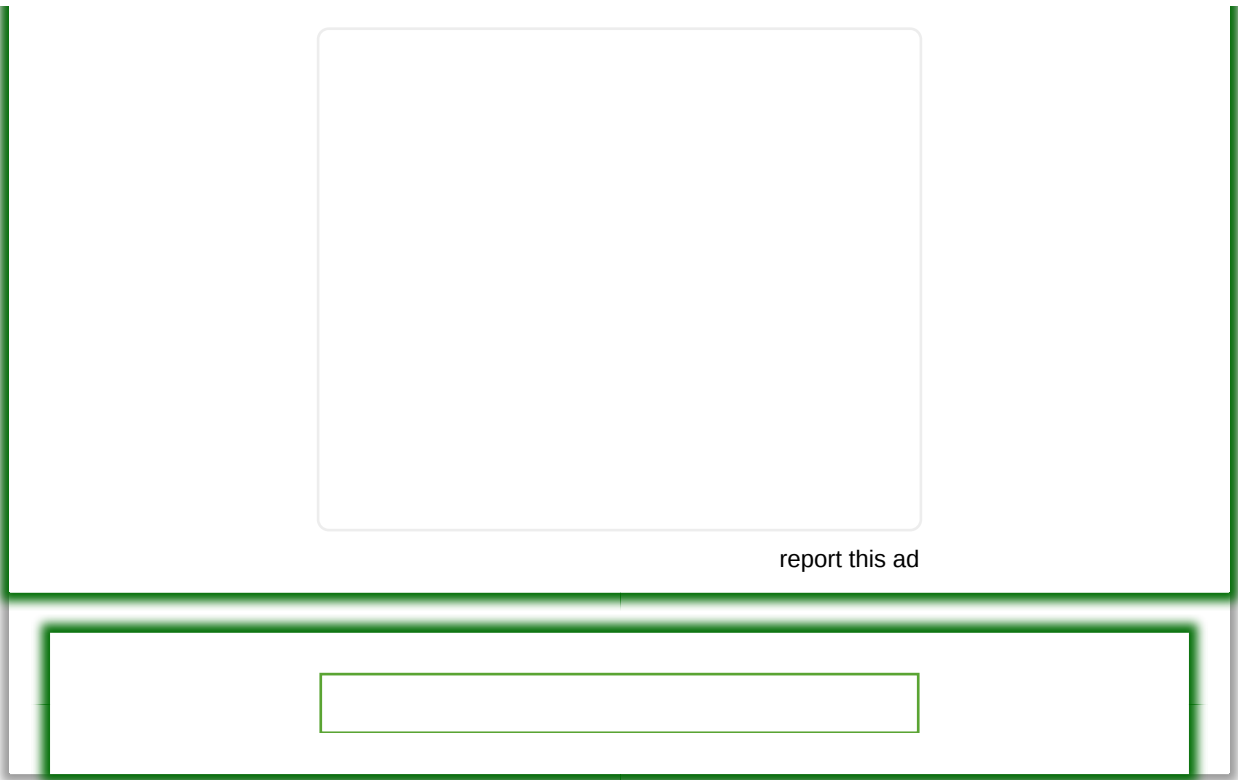
0



Reply

8





U

8

