# Part 3 – Module Parameter

[ <https://embetronicx.com/tutorials/linux/device-drivers/linux-device-driver-tutorial-part-3-passing-arguments-to-device-driver/> ]

# 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.

**Module Parameters Macros**

* module\_param()
* module\_param\_array()
* module\_param\_cb()

Before we 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

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

perm  is the usual permissions value.

## module\_param\_cb()

This macro used to register the callback whenever the argument (parameter) got changed.

For Example,

I have created one parameter by using module\_param().

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 command line by

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 notification.

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

struct kernel\_param\_ops {

int (\***set**)(const char \*val, const struct kernel\_param \*kp);

int (\***get**)(char \*buffer, const struct kernel\_param \*kp);

void (\***free**)(void \*arg);

};

For further explanation, please refer below program.

### When we will need this notification?

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

# 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 string (nameETX).

For module\_param\_array(), i have created one integer array variable ().

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 want get any notification when they got change except the variable which is created by module\_param\_cb() macro.

#include<linux/kernel.h>

#include<linux/init.h>

#include<linux/module.h>

#include<linux/moduleparam.h>

int valueETX, arr\_valueETX[4];

char \*nameETX;

int cb\_valueETX = 0;

module\_param(valueETX, int, S\_IRUSR|S\_IWUSR); //integer value

module\_param(nameETX, charp, S\_IRUSR|S\_IWUSR); //String

module\_param\_array(arr\_valueETX, int, NULL, S\_IRUSR|S\_IWUSR); //Array of integers

/\*----------------------Module\_param\_cb()--------------------------------\*/

int notify\_param(const char \*val, const struct kernel\_param \*kp)

{

int res = param\_set\_int(val, kp); // Use helper for write variable

if(res==0) {

printk(KERN\_INFO "Call back function called...\n");

printk(KERN\_INFO "New value of cb\_valueETX = %d\n", cb\_valueETX);

return 0;

}

return -1;

}

const struct kernel\_param\_ops my\_param\_ops =

{

.set = &notify\_param, // Use our setter ...

.get = &param\_get\_int, // .. and standard getter

};

module\_param\_cb(cb\_valueETX, &my\_param\_ops, &cb\_valueETX, S\_IRUGO|S\_IWUSR );

/\*-------------------------------------------------------------------------\*/

static int \_\_init hello\_world\_init(void)

{

int i;

printk(KERN\_INFO "ValueETX = %d \n", valueETX);

printk(KERN\_INFO "cb\_valueETX = %d \n", cb\_valueETX);

printk(KERN\_INFO "NameETX = %s \n", nameETX);

for (i = 0; i < (sizeof arr\_valueETX / sizeof (int)); i++) {

printk(KERN\_INFO "Arr\_value[%d] = %d\n", i, arr\_valueETX[i]);

}

printk(KERN\_INFO "Kernel Module Inserted Successfully...\n");

return 0;

}

void \_\_exit hello\_world\_exit(void)

{

printk(KERN\_INFO "Kernel Module Removed Successfully...\n");

}

module\_init(hello\_world\_init);

module\_exit(hello\_world\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("EmbeTronicX <embetronicx@gmail.com or admin@embetronicx.com>");

MODULE\_DESCRIPTION("A simple hello world driver");

MODULE\_VERSION("1.0");

# Compiling

This is the code of Makefile.

obj-m += hello\_world\_module.o

KDIR = /lib/modules/$(shell uname -r)/build

all:

make -C $(KDIR)  M=$(shell pwd) modules

clean:

make -C $(KDIR)  M=$(shell pwd) clean

# 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 below picture, every value got passed to our device driver.

…

Now i’m going to check module\_param\_cb() is weather calling that handler function or not. For that i need to change the variable in sysfs.

$ 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 wont get notification.

# Unloading the Driver

Finally unload the driver by using sudo rmmod hello\_world\_module.