Tutorial para o desenvolvimento de módulo USB para o kernel do Linux

Neste tutorial será mostrado como criar e inserir um módulo USB para ler os dados do pen drive inserido. Abaixo é possível visualizar o código fonte do módulo cujo o nome atribuído foi **stick_driver.c**. Na struct usb_device_id pen_table é necessário que se saiba o idVendor e o idProduct do pen_drive que pode ser descoberto inserindo o pen_drive, digitando o comando **lsusb**. Após isso basta inserir os valores em USB_DEVICE no código dentro da struct usb_device_id pen_table.

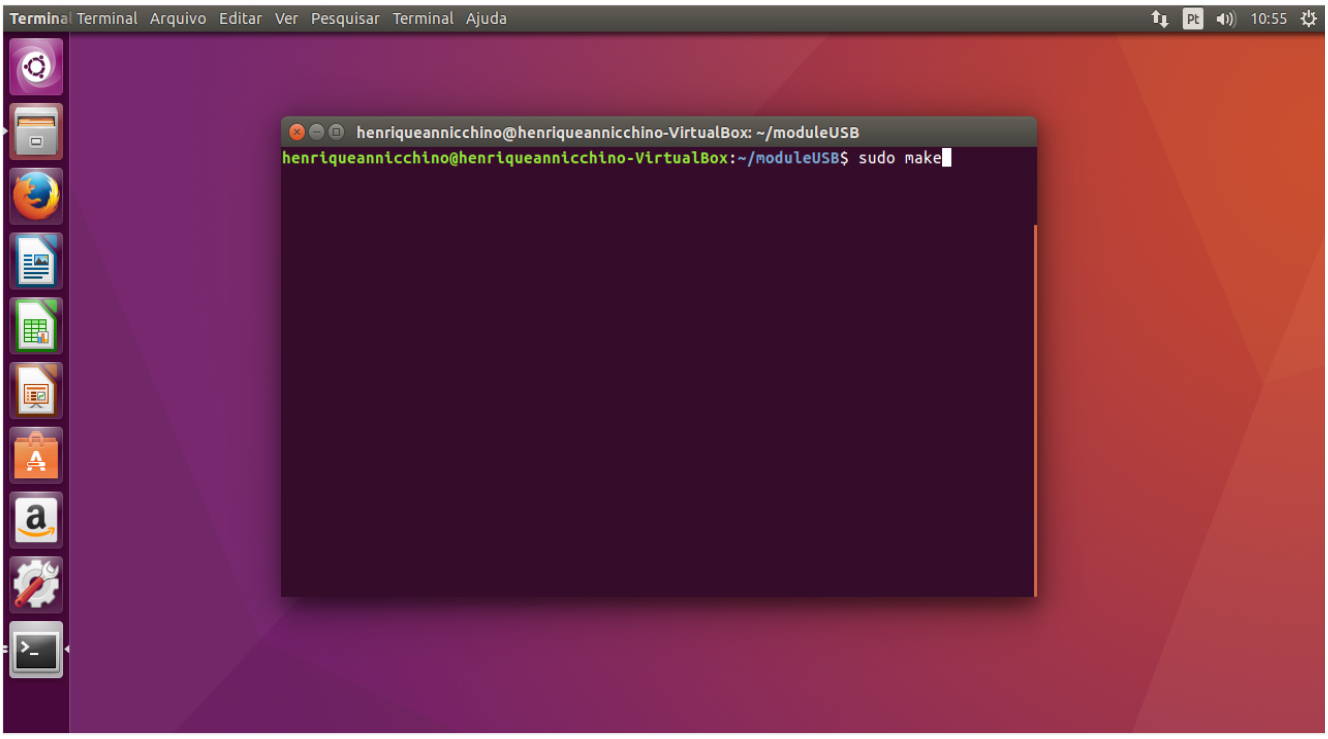
```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/usb.h>


//funcao sonda
//chamada na insercao do dispositivo se e somente se nenhum outro driver nos precedeu
static int pen_probe (struct usb_interface *interface , const struct usb_device_id *id){
        printk(KERN_INFO "[*] Pen Drive (%04X:%04X) Plugged\n" , id->idVendor , id->idProduct);
        return 0;
}

//disconecta
static void pen_disconnect(struct usb_interface *interface){
        printk(KERN_SOH "[*] Pen drive removido\n");
}

//usb_device_id
static struct usb_device_id pen_table[] = {
        //1221:3234 idVendor e idProduct
        {USB_DEVICE(0x1221, 0x3234)}, //informacao obtida usando "lsusb" no terminal
        {} /*entrada terminal*/
};
MODULE_DEVICE_TABLE(usb, pen_table);


// usb_driver
static struct usb_driver pen_driver = {
        .name = "Henrique-USB Stick Driver",
        .id_table = pen_table,//usb_device_id
        .probe = pen_probe,
        .disconnect = pen_disconnect,
};
static int __init pen_init(void){
        int ret = -1;
        printk(KERN_SOH "[*] construtor do driver");
        printk(KERN_SOH "\tRegistrando driver com kernel");
        ret = usb_register(&pen_driver);
```

```c
static struct usb_device_id pen_table[] = {
        //1221:3234 idVendor e idProduct
        {USB_DEVICE(0x1221, 0x3234)}, //informacao obtida usando "lsusb" no terminal
        {} /*entrada terminal*/
};
MODULE_DEVICE_TABLE(usb, pen_table);


// usb_driver
static struct usb_driver pen_driver = {
        .name = "Henrique-USB Stick Driver",
        .id_table = pen_table,//usb_device_id
        .probe = pen_probe,
        .disconnect = pen_disconnect,
};
static int __init pen_init(void){
        int ret = -1;
        printk(KERN_SOH "[*] construtor do driver");
        printk(KERN_SOH "\tRegistrando driver com kernel");
        ret = usb_register(&pen_driver);
        printk(KERN_SOH  "\tRegistro completo");
        return ret;
}

static void __exit pen_exit(void){
        //deregister
        printk(KERN_SOH "[*] Desconstrutor do driver");
        usb_deregister(&pen_driver);
        printk(KERN_SOH "\tregistro cancelado!");
}

module_init(pen_init);
module_exit(pen_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("HenriqueAnnicchino");
MODULE_DESCRIPTION("USB Pen Drive registration");
```

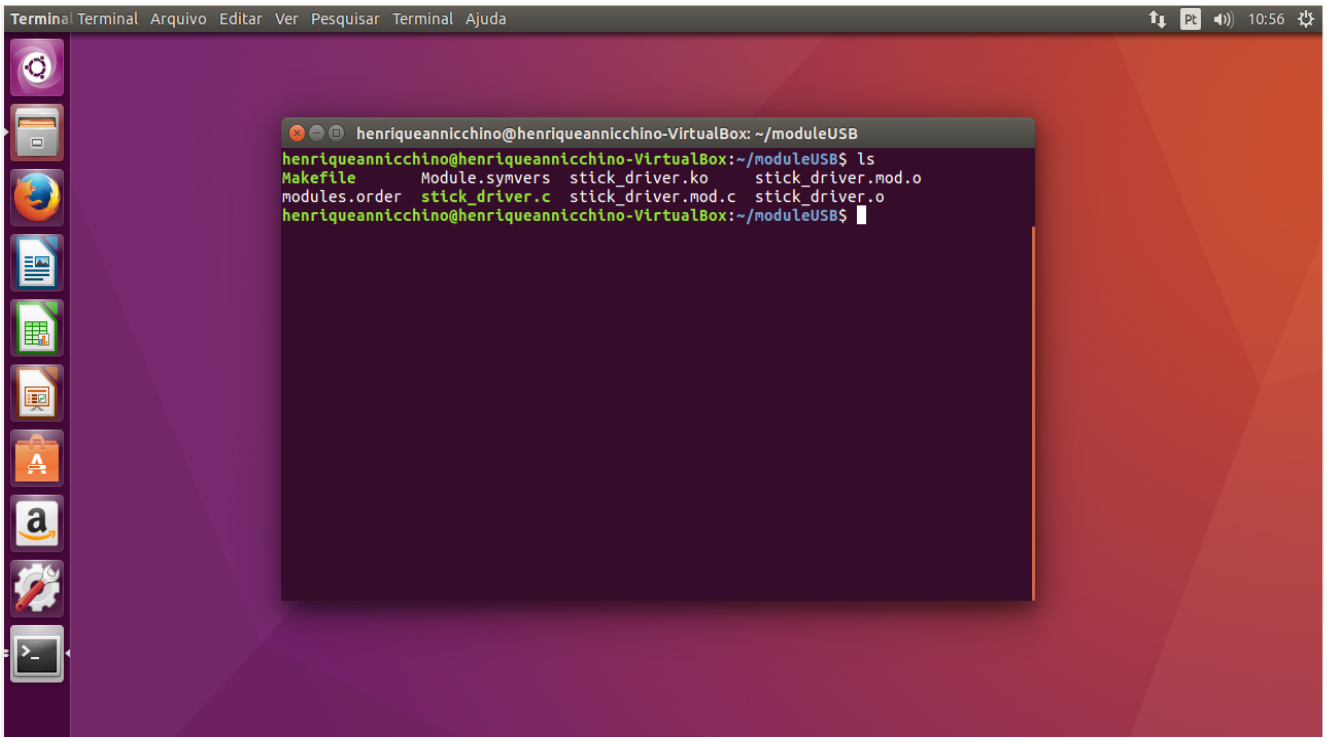
Abaixo é possível visualizar o **Makefile** utilizado.

```
obj-m := stick_driver.o

KERNEL_DIR = /lib/modules/$(shell uname -r)/build
PWD = $(shell pwd)

all:
	$(MAKE) -C $(KERNEL_DIR) SUBDIRS=$(PWD) modules

clean:
	rm -rf *.o *.ko *.mod.* *.symvers *.order *-
```

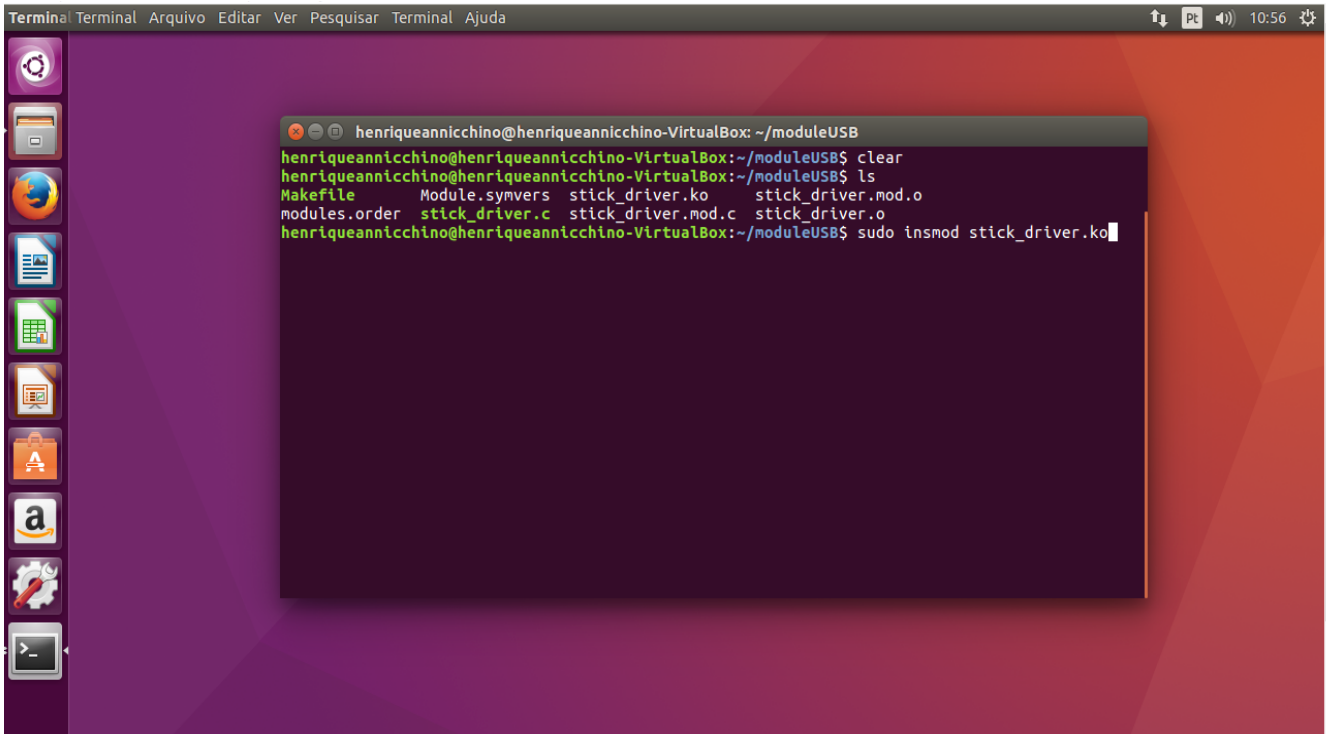
Após já ter criado ambos os arquivos citados acima basta usar o comando **sudo make** no diretório onde os mesmos se encontram.



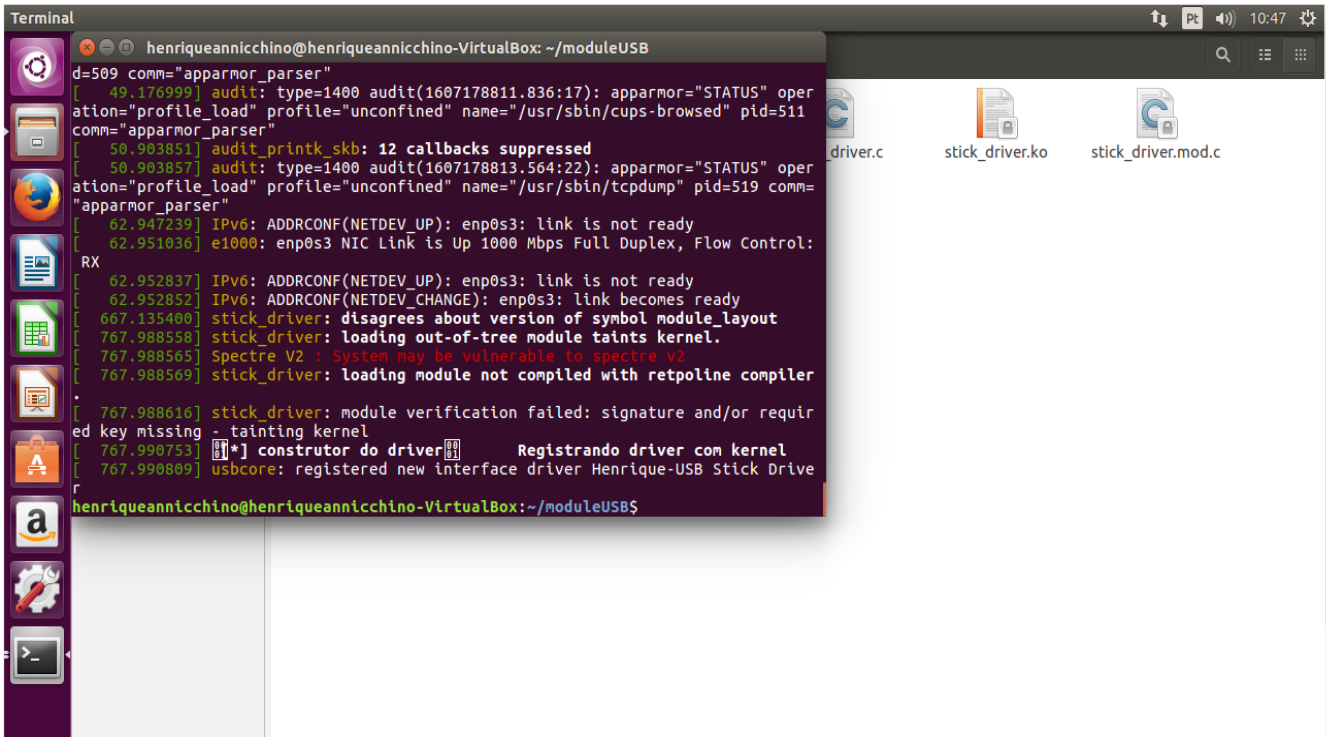
Caso o comando tenha sido executado corretamente será possível mais alguns arquivos no diretório usando o comando **ls** como pode ser visto abaixo.



Para inserir o módulo é necessário utilizar o comando **sudo insmod (nome do arquivo).ko**



Para visualizar se o módulo carregado corretamente basta utilizar o comando **dmesg.**
OBS: Para que o módulo funcione como esperado é preciso remover o módulo usb_storage utilizando o comando **sudo rmmod usb_storage**.



Na imagem abaixo é possível visualizar as mensagens exibidas pela módulo ao inserir, remover e inserir novamente um pen_drive que possua idVendor e idProduct listados no código.