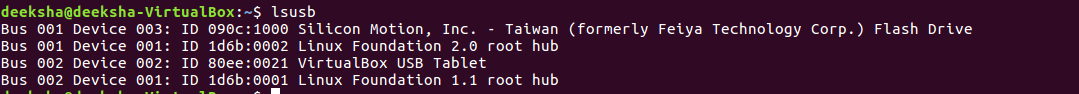
Command to check if the usb is connected : ls /dev/ |grep sd

Every USB device has a vendor ID and major ID. This is how the system recognizes the device. The USB host controller takes all the binary information from the USB and sends to USB core. The USB driver then kernel (OS). Then kernel checks if there is a device driver to handle the device. Then the kernel makes the device available to the user.

The device is configured in such a way that every device has a profile. There are configurations, within these are interfaces. Describes the functionality of the device. Like printer , fax, scanner are the function and each of these have a interface. Each interface has 1 or more end point .Which decribes input and output. Endpoint ) is used to initialize the device. Types of end points : Control, Interrupt , Bulk and Isochronous. All the end points are uni directional except control which is bi directional.

Mass Storage devices use : Bulk and Isochronous

“lsusb” command lists all the USB connected to the device.



Shows vendor and product ID . for my pendrive Product ID :1000 vendor ID : 090c .This is needed in programming the device.

USB driver :

Have a structure which is the basic definition of USB.

ID table is used to match this driver with the device.

#include<linux/module.h>

#include<linux/kernel.h>

#include<linux/usb.h>

// probe function

// called on device insertion if and only if no other driver is using the device

Static int pen\_probe(struct usb\_interface \*interface, const struct usb\_device\_id \*id)

{

Printk(KERN\_INFO “[\*] pendrive (%04X : %04X) plugged \n”, id->idVendor, id->idProduct);

Return 0; //returning 0 will indicate that we will manage the device

}

// disconnect function

Static void pen\_disconnect(struct usb\_interface \*interface){

Printk(KERN\_INFO “[\*] pendrive has been removed”);

}

//usb\_device\_id

Static struct usb\_device\_id pen\_table[]={

//0781:5406

{USB\_DEVICE(0x0781, 0x5406)}, //information used from lsusb command, vendor ID, product ID

{} // terminating entry

};

MODULE\_DEVICE\_TABLE (usb, pen\_table);

//usb driver

Static struct usb\_driver pendrive \*

{

.name=”Transend USB”;

.id\_table=pen\_table; // used to match this driver with the any device which is attached to usb bus

.probe = pen\_probe; // probe function

.disconnect = pen\_disconnect; // clean up memory

};

// initializing the pen drive to the USB core

Static int \_\_init pen\_init(void){

Int ret = -1;

Printk(KERN\_INFO “[\*] pendrive constructor of driver”);

Printk(KERN\_INFO “ \t Register Driver with kernel”);

Ret= usb\_register(&pen\_driver);

Printk(KERN\_INFO “ \t Registeration is complete”);

Return ret;

}

Static void \_\_exit pen\_exit(void){

Printk(KERN\_INFO “[\*] pendrive destructor of driver”);

Usb\_deregister(&open\_driver);

Printk(KERN\_INFO “\t unregisteration complete “);

}

Module\_init(pen\_init);

Module\_exit(pen\_exit);

MODULE\_LICENSE(“GPL”);

MODULE\_AUTHOR(“pendrive”);

MODULE\_DESCRIPTION(“USB Pen Registration Driver”);

MAKE file :

Obj-m := stick\_driver.o

KERNEL\_DIR = /lib/modules/$(shell uname -r)/build // directory to the source file of the module

PWD = $(shell pwd)

all:

$(MAKE) -C $(KERNEL\_DIR) SUBDIRS=$(PWD) modules

clean:

rm -rf \*.o \*.ko \*.mod \*.symvers \*.order \*-