IPASJ International Journal of Information Technology (IIJIT)
Web Site: http://www.ipasj.org/IIJIT/IIJIT.htm
A Publisher for Research Motivation ....
Email:editoriijit@ipasj.org
Volume 5, Issue 1, January 2017
ISSN 2321-5976

A Frame job for practical USB Devices under Linux atmosphere

Mr. Chirag Bhadotriya

Bidhan Chandra Krishi Viswa Vidyalaya, Nadia

ABSTRACT

In trade computer code development for USB devices is extremely stringent. the target of this analysis work is to develop a framework for writing and testing the computer code for USB devices even before the devices ar factory-made. Faulty hardware is usually a bottleneck in computer code development and time is usually lost thanks to this perplexity. there's want for a framework that would dependably take a look at the computer code while not the necessity of actual hardware. Minimizing the time to sell a product is that the key to success in today's growing market. With the event of this framework, hardware and software system development are often exhausted parallel. Such a frame work are often developed on any software package like windows, Linux etc. during this analysis work, we've got elect Linux software package for the implementation of Virtual Device Frame work thanks to its open supply nature.

1. INTRODUCTION

THE Universal Serial Bus (USB) may be a quick and versatile interface for connecting devices to computers. each new laptop has a minimum of a few of USB ports. The interface is flexible enough to use with commonplace peripherals like keyboards and disk drives still as additional specialised devices, together with one-of-a-kind styles. In short, USB is extremely totally different from the heritage interfaces it's substitution. A USB device [8] could use any of 4 transfer sorts and 3 speeds. On attaching to a laptop, a tool should reply to a series of requests that change the laptop to be told regarding the device and establish communications with it. In the PC, each device should have a low-level driver to manage communications between applications and also the system's USB drivers [9]. Developing a USB device and also the software system that communicates with it needs knowing one thing regarding however USB works and the way the PC's software package implements the interface. USB communications takes place between the host and endpoints placed within the peripherals. Associate in Nursing terminus may be a unambiguously available portion of the peripheral that's the supply or receiver of information. Four bits outline the device's terminus address; codes conjointly indicate transfer direction and whether or not the dealing may be a "control" transfer. terminus zero is reserved for management transfers, feat up to fifteen bi-directional destinations or sources of information inside every device [1].

The idea of endpoints ends up in a vital thought in USB transactions, that of the pipe. All transfers occur through virtual pipes that connect the peripheral's endpoints with the host. once establishing communications with the peripheral, every terminus returns a descriptor, a knowledge structure that tells the host regarding the endpoint's configuration and expectations. Descriptors embody transfer kind, liquid ecstasy size of information packets, maybe the interval for information transfers, and in some cases, the information measure required. Given this information, the host establishes connections to the endpoints through virtual pipes, that even have a size (bandwidth), to form them analogous to house plumbing [3]. USB supports four information transfer types: management, isochronal, bulk, and interrupt. management transfers exchange configuration, setup, and command data between the device and also the host. CRCs check the info and initiate retransmissions once required to ensure the correctness of those packets. Bulk transfers move giant amounts of information once timely delivery is not vital. Typical applications embody printers and scanners. Bulk transfers ar fillers, claiming unused USB information measure once nothing additional vital goes on. CRCs shield these packets. Finally, isochronal transfers handle streaming information like that from Associate in Nursing audio or video device. it's time sensitive data thus, inside limitations, it's warranted access to the USB bus. No error checking happens therefore the system should tolerate occasional disorganised bytes [6], [7].

In trade computer code development for USB devices is extremely stringent. the target of this project is to develop a framework for writing and testing the computer code for USB devices even before the devices ar factory-made. Faulty hardware is usually a bottleneck in computer code development and time is usually lost thanks to this perplexity. there's want for a framework that would dependably take a look at the computer code while not the necessity of actual hardware. Time is cash. Minimizing the time to sell a product is that the key to success in today's growing market. With the event of this a framework, hardware and software system development are often exhausted parallel. Such a frame work are often developed on any software package like windows, Linux [5] etc. during this analysis we've got elect Linux software package [4] for the implementation of Virtual Device Frame work. the most reason for this choice is that the open supply nature of Linux. conjointly there ar several categories for USB devices. one in all them is Mass Storage category [12]. Currently, the scope of project is restricted to USB Mass Storage category solely. the remainder of the paper is organized as: In section II, we tend to mentioned the background history and connected work of the analysis. In section III, we tend to given our planned system style and design. we tend to conclude our add section IV.

2. BACKGROUND

A. The USB Mass Storage category

This section offers summary|an summary|an outline} of the USB Mass Storage category [12] specification overview. however mass storage devices behave on the USB bus is that the subject of this and different USB Mass Storage category specifications. additionally to the present summary specification, many different USB Mass Storage category specifications ar supported by the USB Mass Storage category unit (CWG).

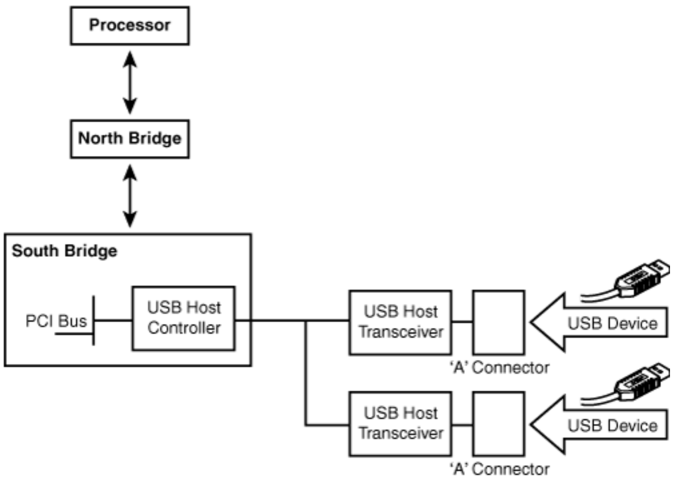

Fig:-1 USB Hub Architecture

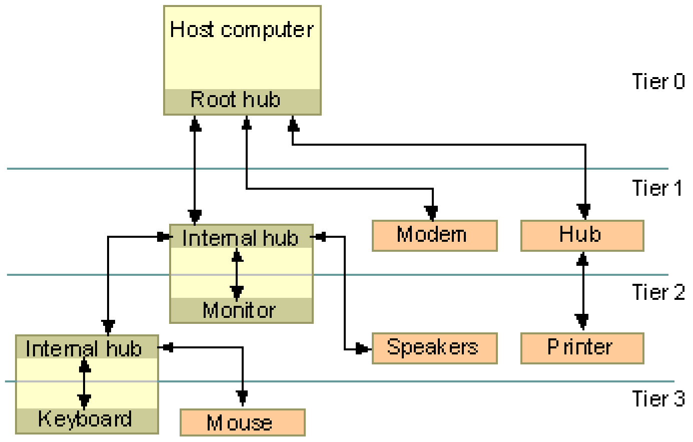

Fig:-2 Linux USB Architecture

CBI Transport specification is approved to be used solely with full-speed disc drives. CBI shall not be employed in highspeed capable devices, or in devices apart from disc drives. Usage of CBI for any new style is discouraged. Note: The bootability and Compliance take a look at specifications ar still underneath development, and don't seem to be however in public on the market.

1). Specification Relationships

The CBI and Bulk-Only specifications ar every meant to be complete documents for the USB Mass Storage category, sanctionative development of a USB Mass Storage compliant device. a tool manufacturer could opt to implement each

CBI and Bulk-Only, however shall follow every specification as applicable. Booting Associate in Nursing software package from a USB Mass Storage category device needs no special issues with respect to Mass Storage category support. Either CBI or BulkOnly devices is also bootable. Bootability could, however, need different issues like explicit forms of media format, etc. Such issues ar hardware- or software package dependent, and ar on the far side the scope of the Mass Storage category specifications.

2). Mass Storage taxonomic category
The Interface Descriptor of a USB Mass Storage category (Fig. 2) device includes a bInterfaceSubClass field. This field denotes the industry-standard protocol transported by a Mass Storage category interface. the worth of the bInterfaceSubClass field shall be set to 1 of the taxonomic category codes as shown within the following table. Note that the taxonomic category code values employed in the bInterfaceSubClass field specify the industry-standard specification that defines transport protocols and command code systems transported by the interface; these taxonomic category codes don't specify a sort of device (such as a compact disc read-only memory or disc drive).

B. Linux USB design
In Linux there exists a scheme known as ``The USB Core" with a selected API to support USB devices and host controllers. Its purpose is to abstract all hardware or device dependent elements by shaping a collection of information structures, macros and functions. The USB core contains routines common to all or any USB device drivers and host controller drivers. These functions are often sorted into Associate in Nursing higher and a lower API layer. There exists Associate in Nursing API for USB device drivers and another one for host controllers. the subsequent section concentrates on the USB utility layer, as a result of the event for host controller drivers is already finished. This section can provide an summary of the USB framework by explaining entry points and also the usage of API functions. The Fig. three shows the design of Linux USB [2], [5], [14].

1). Framework information Structures
USB devices drivers ar registered and deregistered at the scheme. A driver should register two entry points and its name. For specific USB devices (which don't seem to be appropriate to be registered at the other subsystem) a driver could register a few of file operations and a minor range. during this case the desired minor range and also the fifteen following numbers ar assigned to the motive force. This makes it attainable to supply to sixteen similar USB devices by one driver. the main range of all USB devices is one hundred eighty.

3.PLANNED SYSTEM STYLE

A. File-backed Storage
The File-backed Storage (FS) provides support for the USB Mass Storage category. It will seem to a number as a collection of up to eight SCSI disk drives (called Logical UNits or LUNs), though most of the time one LUN is all you'll want. the data keep for every LUN should be maintained by the contrivance somewhere, either in a very traditional file or in a very block device like a disk partition or maybe a ramdisk.

Under Linux two.6 [4], [15]; if "removable=y" is side to the modprobe line then FSG can act sort of a device with removable media and permit to specify the backing storage victimization sysfs attributes. In fact, by doing this the "file=..." parameter are often omited entirely. The contrivance can gibe a zipper drive with no cartridge inserted till sysfs is employed to specify some backing storage. a vital WARNING! whereas FSG is running and also the contrivance is connected to a USB host, that USB host can use the backing storage as a personal disc drive. it'll not expect to visualize any changes within the backing storage apart from those it makes. Extraneous changes ar prone to corrupt the filesystem and will even crash the host. just one system (normally, the USB host) could write to the backing storage, and if one system is writing that information, no different ought to be reading it. the sole safe thanks to share the backing storage between the host and also the gadget's software package at constant time is to form it readonly on each side.

B. Partitioning the Backing Storage
However, making the backing storage is not enough. It's like having a raw disk drive; that wants partition and filing system before victimization it. (Strictly speaking there's no ought to partition it. the whole drive are often treated as one giant device, sort of a disc. this may be confusing, though, and a few versions of Windows will not work with Associate in Nursing united USB drive). To partition the backing storage a partition table must be created by victimization the fdisk program. Here's Associate in Nursing example showing a way to make out. the instance assumes that contrivance are going to be used with a Windows host. it is a very little difficult as a result of fdisk wants facilitate once operating with one thing apart from Associate in Nursing actual device. Begin by kicking off fdisk and telling it the name of

backing storage. A message one thing like this: bash# fdisk /root/data/backing_file are going to be received. Device contains neither a sound DOS partition table, nor Sun or SGI disklabel. Build a brand new DOS disklabel. Changes can stay in memory solely, till you opt to put in writing them. After that, of course, the previous content will not be retrievable.

D. Heads, Sectors and Cylinders As fdisk has to set the heads, sectors, and cylinders values. (Some versions solely ought to set the quantity of cylinders, however they are wrong. it's baecause of fault of the dimensions of backing file; as default values ar used and ignoring the particular file size.)
The numbers ar somewhat arbitrary; the theme shown here works sensible. provide the "x" (eXpert or eXtra) command: Command (m for help): x
Then range of sectors/track are going to be set. g_file_storage uses a sector size of 512 bytes, thus eight sectors/track can provide 4096 bytes per track. this is often sensible as a result of it matches the dimensions of a memory page (on a 32-bit processor). knowledgeable command (m for help): s range of sectors (1-63): eight
Warning: setting sector offset for DOS compatiblity Next is to line the quantity of heads (or tracks/cylinder). With four K per track, sixteen heads can offers a complete of sixty four K per cylinder, that is convenient since the dimensions of the backing file is sixty four MB. knowledgeable command (m for help): h
Number of heads (1-256): sixteen Finally the quantity of cylinders are going to be set. it is vital that the full size ought to match the particular size of the backing file. Since there ar sixty four K per cylinder and sixty four MB total, 1024 cylinders ar required. knowledgeable command (m for help): c

E. making a Primary Partition
Create a brand new primary partition ("n" for new). build it no 1. The defaults for the beginning and ending cylinder ar good as a result of they'll build the partition occupy the whole.
Command (m for help): w The partition table has been altered! occupation ioctl() to read partition table. Re-read table unsuccessful with error 25: Inappropriate ioctl for device. boot system to confirm the partition table is updated.
WARNING: If you have got created or changed any DOS half-dozen.x partitions, please see the fdisk manual page for extra data.

F. Adding a filing system
At this time a brand new partition has been created however it does not however contain a filesystem. the simplest thanks to add a filesystem is to load g_file_storage, connect the contrivance to a USB host, and use the host to try to to the work. With a Linux host [13] run mkdosfs; with a Windows host. Or double-click on the drive's icon within the "My Computer" window.

G. Accessing the Backing Storage from the contrivance
It is attainable to control the info within the backing storage from the contrivance (even to feature the filesystem). do not do that whereas the contrivance is connected to a USB host! The key's to use the loop utility with the "-o" (offset) choice for the losetup program. For this to figure, confirm the partition's offset. Following the theme given on top of would end in 4096. If not, one will use fdisk to search out the right offset value: # fdisk -lu /root/data/backing_file One should set cylinders.

4. CONCLUSIONS AND FUTURE ENHANCEMENTS

We have shown that virtual USB thought work well with mass storage category. this may prove itself an excellent testing surroundings as so much as testing of USB drivers is bothered. Mass storage category was tested nearly victimization this framework and it proved to be an honest operating surroundings even once the particular device is underneath development part. USB Host facet drivers are often developed while not the necessity of any real device. This approach may also facilitate in testing mass storage devices of various capacities nearly.

REFERENCES

[1]. Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips. Universal Serial Bus Specification - Revision two.0, April 2000. http://www.usb.org.
[2]. Detlef Fliegl. Programming Guide for Linux USB Device Drivers - Revision one.32, 2000. http://usb.cs.tum.edu/usbdoc
[3]. SystemSoft and Intel. Universal Serial Bus Common category Specification – Revision one.0, Gregorian calendar month 1997. http://www.usb.org/developers/devclass_docs.

[4]. Linux kernel sources. http://www.kernel.org/.
[5]. Linux USB sources http://www.linux-usb.org/
[6]. Intel. Universal Host Controller Interface (UHCI) style Guide - Revision one.1, March 1996. http://www.usb.org/developers/docs/
[7]. Intel. increased Host Controller Interface for Universal Serial Bus - Revision one.0, March 2002. http://www.usb.org/developers/docs/
[8]. Intel USB supply http://www.intel.com/technology/usb/
[9]. Wikipedia USB supply http://en.wikipedia.org/wiki/USB.
[10]. Wiki USB supply, http://www.en.wikipedia.org/wiki/USB_flash_drive/
[11]. Linux USB supply, www.linux-usb.org/USB-guide/x498.html
[12]. Mass Storage USB supply World Wide Web.gentoowiki.com/HOWTO_USB_Mass_Storage_Device
[13]. Linux USB supply, www.linux.about.com/od/linux101/a/desktop04c.htm.
[14]. Linux USB sources, www.gsi.de/informationen/wti/library/scientificreport2006/PA PERS/INSTRUMENTS-METHODS-15.pdf
[15]. Linux USB supply http://www.linux-usb.org/.