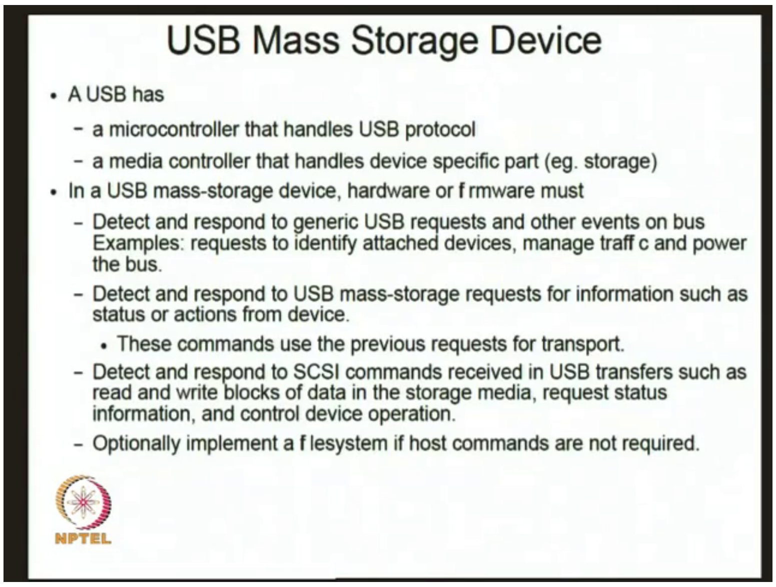
Storage Systems
Dr. K. Gopinath
Department of Computer Science and Engineering
Indian Institute of Science, Bangalore

Types of Storage Devices and Systems, Long-term Storage
Lecture – 11
USB Storage: Introduction, Layers, Linux USB Framework, Core APIs

Welcome again to the NPTEL course on storage systems. Today we will look at one widely used storage medium slash storage system ok.

(Refer Slide Time: 00:35)



USB Mass Storage Device

- A USB has
  - a microcontroller that handles USB protocol
  - a media controller that handles device specific part (eg. storage)
- In a USB mass-storage device, hardware or f rmware must
  - Detect and respond to generic USB requests and other events on bus Examples: requests to identify attached devices, manage traff c and power the bus.
  - Detect and respond to USB mass-storage requests for information such as status or actions from device.
    - These commands use the previous requests for transport.
  - Detect and respond to SCSI commands received in USB transfers such as read and write blocks of data in the storage media, request status information, and control device operation.
  - Optionally implement a f lesystem if host commands are not required.

What is that one? That is the USB mass storage device. We just go through it a bit it has got a lot of interesting aspects and many of this aspects are there in any other two system typical also. You look at it carefully we will find similar aspects, that is why it is interesting to look at it.

So, I am pretty sure that most of you have used it right. It is widely available and it came out first about 2000. Now and if first came out 64 megabyte was considered big, 32 megabytes considered big now it is close to 64 gigabytes can we get. Somebody actualize now there is somebody has come with a 1 terabyte USB stick. So, it is the progress in this when comes to density has been phenomena and what is the technology that is fueling it. It is something called flash storage. Flash storage is a type of storage which is personal type of memory. It is similar to semiconductor memory, but it has got some differences we will have going to details about those thing because its not part of our discussion here ok.

So, it essentially semiconductor storage; essentially there is some kind of a potential we will that is created under trap some electrons. Whether electrons are there are not that basically says 0 or 1 and because it is an electronic technology, it does not have mechanical components. So, some of the issues that are there with the disks are not there, but in spite of that it turns out the electro mechanical technology still in terms of density much better than what we can do with flash ok.

For example you might have seen that 4 terabyte hard disks are available, 3 terabytes are easily available, 4 terabyte you can face spend some effort you will find it, but getting this flash at that the same plus point is still not easy. Flash is about a factor of 10 times costlier than disk. So, anyway say tells out this us there is a based on the flash to flash kind of semiconductor technology, you can build a mass storage device using USB protocol.

What is USB protocol? It is basically a way of connecting some functionality in the form of some device to the some system. It could be a PC, it could be a laptop, it could be a camera, it could be a printer etcetera right there are so many types of technologies, this basically the USB protocol is a way of having a electrical connection on top of it a communication let us say protocol on top of it so that some of the transfers that are required for the device to operate with the camera or the PC can be made possible.

Let us quickly look at what is there in the systems. USB has a micro control that handles USB protocol. Again we will go a bit into more detail, but the handles USB protocol and media controller that handles device specific part example storage in the storage. As I told could be different kinds of USB devices. One could be doing storage, one could be doing Wi-Fi also right, one could be doing a printer also. So, there is part which is connected to the USB protocol, the part is connected the device part of it. For example, the storage will be it is own for example; it is how something called SCSI commands SCSI. S C S I ok.

Now, that is specific to a storage device, but it is not what a Wi-Fi device, that is not necessary that is something different right. So, if you are talking about the USB mass storage device, what are we doing right now? You are basically we take the device an insert it into a laptop or a PC. The question is what happens we will slightly taken look at this one.

First of all usual to detect and respond to generic USB requests and other events on bus because something on the bus; because an electrical system I put it in and then there is an interrupt generated the electrical connection and basically once that happens then I need to know I need to tell the device now putting a information on the bus I want to know who you are. So, requests identifies at devices and this information is in the device itself, usually in a non-volatile part of the device so that it can it does not have to be lost anytime you remove power And also there are some specific things to respect to traffic, means whether it is whether data should be going from this side to that side from the PC side to the USB or USB side to the PC etcetera and also the power in the bus. I think all of you know that the USB device the storage device have does not have power of its own. So, it has to be powered by the PC it goes to the sock the USB socket and that power is given to it right and there are this power in the bus also happens to that let us say there is a specific aspect of how to power it also. So, detect and respond USB mass storage request, one is the generic USB requests, this one is the USB mass storage requests such as status or actions from device.

For example I might want to know please are you ready for example (Refer Time: 06:57). So, for example, it may be that I need some information about the status of some previous action (Refer Time: 07:09) and since it is a USB mass storage, it turns out they decided use something called SCSI commands. We will go a bit into it later, but basically SCSI commands are a way of interacting with storage devices. It is become the most popular protocol for interacting with storage devices and this SCSI commands or things like read from particular block, write from particular write a particular block and it could even things like retry the previous request because I could not you send me something, but I cannot make sense of it. Keep the information about your previous whatever happened last time, I want to know the status of that one (Refer Time: 07:57) this SCSI protocol has command of that kind and such as read and write blocks of data in the storage media, request status information, control device operation and these are the SCSI level, there is also one more thing at the file system level ok.

For example, you know you will notice that most of the USB devices are formatted to something called FAT32 file system. That means, it has got a particular notion of how the, let us say the file system metadata where is it kept. So, and then since you know where the file system is metadata is there, you can do what is called a mounting the device. So, let us accesses were the file system. What is it mean to say that the device accessible of file system. In a UNIX kind of system this particular device now is in the name space of the UNIX system. For example, I might have it as slash dev USB on a lying system and then I can CD to that slash dev slash USB whatever slash dev slash whatever and you can change the directory do that, and I will it would just look like to look to me like a regular directory once it is mounted ok.

Again what is the mounting operation? You have a raw device, you insert it into the device whatever and then some control actions take place and then you finally, draft the file system that is sitting on the device, onto the that is attach it to at some point to the UNIX file system, and then from then onwards you can change directory to that place and we can just do we can navigate up and down the file system structures just like a regular directory or a regular system.

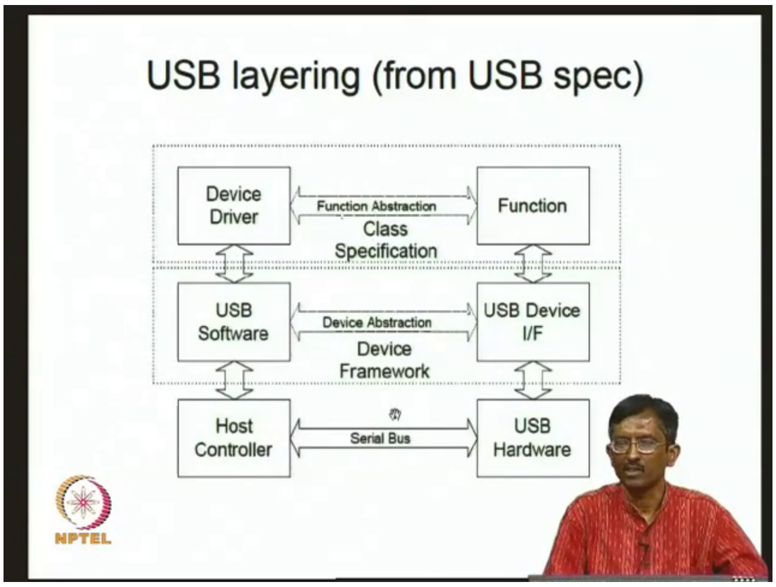
So, and you will notice that if you are having a camera for example, the assumption is that the files are stored as in FAT32 form everything is using FAT32. So, that you can incorporate some intelligence in the camera itself, that is the FAT32 file system and you can start using it. The way a FAT32 system you get. Some other file system that is a big problem, somebody has to make sure that that particular driver for that particular file system is there. That kernel the (Refer Time: 10:50) the device driver for that file system is there.

So, what is happened is that people have assumed that FAT32 is a common denominator everybody can assume that itself. Now, that is independent of your accessing things at SCSI level, the file systems are no (Refer Time: 11:08) level high. For examples, SCSI only talks in terms of blocks. It does not have the notion of directories etcetera. But the file system has a notion of directories. It also has a notion that this if I if there is a directory, I can talk about parent of the directory. There is in some sense I can have a tree like structure. A SCSI command other hand does not SCSI protocol has no notions of parent of a particular piece of formation, it does not have this kind of information.

Basically in some sense directories are not there in SCSI. So, what is useful is a file system it basically I was said to name things and retrieve it using easily remembered names. That is what a file system do, its suddenly much more complicated and what I am saying right now, because when it comes to concurrency and caching and so many things its quite delicate it is not easy to get in making. It is very difficult to get all this is working very well. But impossible what is doing is, we are giving assigning names to piece of information and be able to access it later ok.

In a directory in a tree structured form. So, you can see the multiple levels at which you are operating. One is at the level of hardware with basically senses the device; second thing is the transfer information across the PC and the device; for example that the some kind of communication infrastructure is there. Then you need to able to look at this information that at the block level that it is basically SCSI command level and if you want it to be even more intuiting, you want access it at the level of files then it one more level. So, there are multiple levels at which you are operating.

(Refer Slide Time: 13:00)



USB layering (from USB spec)

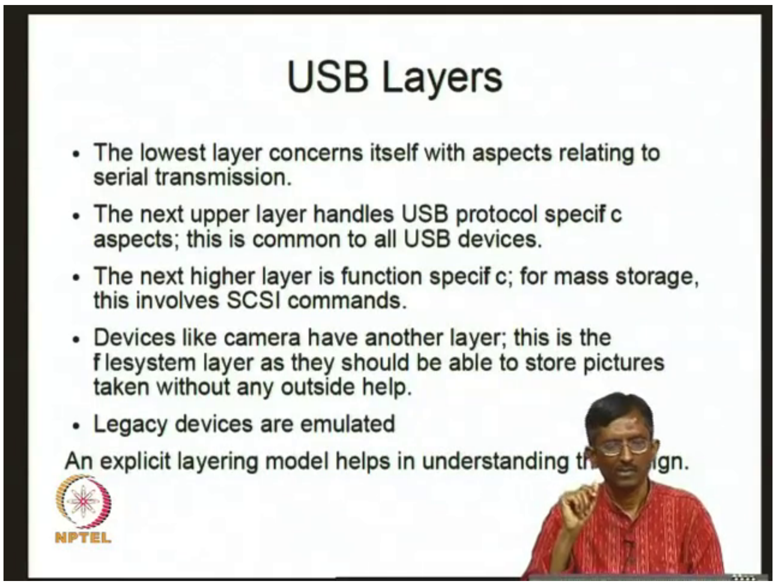
So, I just picked up this picture from the USB specifications the diagram that draw a something similar to this. Let us see what is this? So, there is a USB hardware here and there is a host controller and then there is a serial bus which is connected to this and basically there is a device driver in the kernel it has got a function of abstraction of what it is doing. For example, this device driver is for let us say it assumes that there is a capability for accessing certain blocks, some name blocks that is what is assuming and this host controller. What is it doing? Its basically preparing for data to be transferred on the serial bus it is noting when the transfer is complete. It also good for example, be a DMA. It can do DMA access so, all this part of it is here ok.

So, there is also a device abstraction basically, you have lot of interfaces, a good example is suppose you have a camera. Sorry let us take a instead of camera let us take a printer. The printer has the functionality of printing and also sometimes scanning, faxing etcetera. That have various interfaces so, the question is which kind of interface you are referring to all those things come into picture here so, sometimes there is only one interface that simple. But sometimes you can have multiple interfaces and you have to decide to choose which interface. For each interface you might want to consider the device also in different ways. Remember printer its not require certain configurations which is different from what is required for scanning all that is handled here.

So, the sense your applications are somewhere here they are accessing it through some file system and then finally, the file system access the device driver and then that in turn goes through this part and then comes up this. In a sense there is a connection between the device driver is talking to the USB at a functional level. The host controller is talking to USB hardware at the electrical level or bit level you can call it bit level because nowadays I can even do the following. I have what is called the think client right and I want to put the USB device on my think client, now think client is talking to server somewhere else.

I actually wanted to make the server pretend as if it is the USB is connected to the server. Right somebody is going to do some faking out here. Instead of a serial bus which is happening at the electrical bit level. It might be happening at the bit level, but using network that also is possible so, in a sense you will see that there are different levels abstraction in the system and you can understand this whole thing only if you understand these levels. We will go at bit into more detail some detail further detail as we go along.

(Refer Slide Time: 16:47)



USB Layers

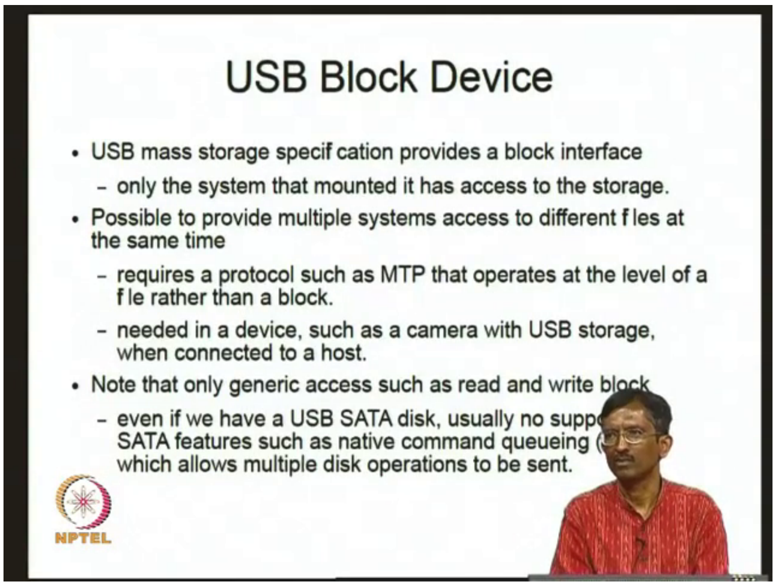
- The lowest layer concerns itself with aspects relating to serial transmission.
- The next upper layer handles USB protocol specif c aspects; this is common to all USB devices.
- The next higher layer is function specif c; for mass storage, this involves SCSI commands.
- Devices like camera have another layer; this is the f lesystem layer as they should be able to store pictures taken without any outside help.
- Legacy devices are emulated

An explicit layering model helps in understanding t[illegible]ign.

So, again to repeat what are the USB layers. The lowest layer concerns itself with aspect relating to serial transformation. The next upper layer handles USB protocol specific aspect; which is common to all USB devices. The next layer is function specific for mass storage this involves SCSI commands. So, devices like camera have another layer, this is the file system layer as they should be able to store picture taken without any outside help because you do not need of PC for that. We should able to do it itself so, it has also that this is might be call it the file system layer and might even have a camera might even have some other higher-level things on top of it.

It may be also be that you can attached legacy device which USB support. Alright, we already talked about if you want to keep data alive for a century it is not revile. So, what I want to this is I have a floppy now I want to read it 100 years from now. What can I do? I want to emulate the legacy device the device of 1990. I want to emulate in 2015, but I want to use USB protocol for it. That is all somehow figure out a way of connecting that particular floppy device, floppy disk I will electrically do something interesting. So, that it can be some off it in to the I can connected to a USB let us say type connection USB type device suppose. I am able to do it, but still there is a piece of software that has to be there right so, that has to be also emulated.

(Refer Slide Time: 18:43)



USB Block Device

- USB mass storage specif cation provides a block interface
  - only the system that mounted it has access to the storage.
- Possible to provide multiple systems access to different f les at the same time
  - requires a protocol such as MTP that operates at the level of a f le rather than a block.
  - needed in a device, such as a camera with USB storage, when connected to a host.
- Note that only generic access such as read and write block
  - even if we have a USB SATA disk, usually no supp SATA features such as native command queueing (which allows multiple disk operations to be sent.

Now, USB mass storage specification provides that is call a block interface. What is it mean of block interface? It means it is accessing things are the in units of blocks it is not at a at a byte level, it is at a block level block level. Block level can be 4 kilobyte blocks, 25, 12 byte blocks whatever that is up to you ok so, there is a block size it is accessing it in only in terms of blocks.

You know it is a block because it is a block interface you need to go through a protocol called mounting. A mounting protocol essentially is a initialization step before you can access the block device. Basically, what we are doing is we have the information on the USB device and you want to graft it, you want to connect it into the into the file system tree that is what you are going to do.

So, once you do the mounting that part of it is done that is you are able to take the file system that sitting on the USB device and you are now able to attach it to your main computer systems file system attaching it some particular place so, that all the files that are there on the device are now accessible from the machine, from the PC and you can as I mentioned before you can change the directory to anyplace and you can walk up and down the file system tree so, the only system there is mounted access again thing is if you are mounting it.

The party you has a mounted it has a exclusive control on it. Basically, the idea is that before concurrent request come in there is a problem right because if there are concurrent requests one guy can leave this you sometimes have to modified things. In a modified things what happens is it might have touch three different places before your operations complete.

Now, you might have done two things before you do a third one somebody else concerned does something else. The system is in an inconsistent state. Somebody else is able to watch sees things in a state before all the operations are complete which is can be serious problem so, you want to avoid it so, if you want to one way to do is to mount it. And therefore, only allow one person there are some times you need multiple parties to access it will be same time so, you might be an another protocol for that that is not standard ok.

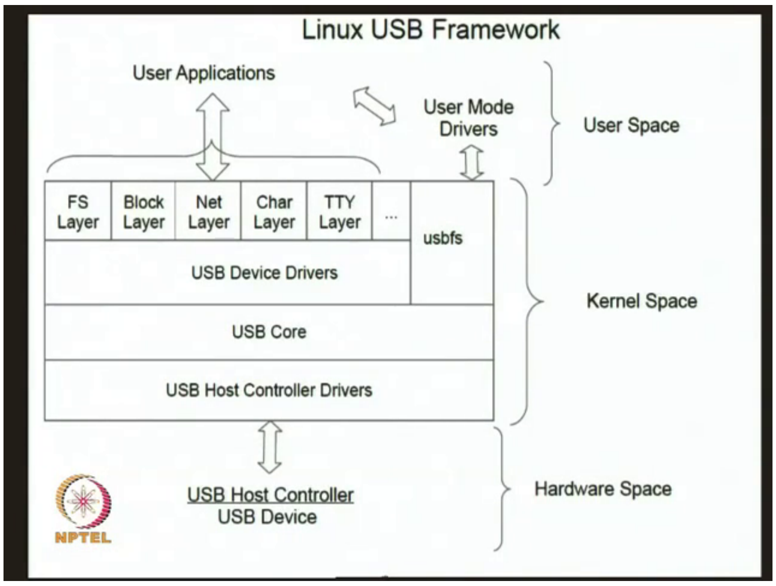
So, for example, camera with USB storage or even if you take one of these android devices you attach it to USB port. It gives a two options right it says do you want to think of me as USB device or the storage device and a somebody else right. If it is used as a USB storage device then somebody has to program that particular android device as a bits only a mass storage device and that has to be done by the mass arrangement call it the PC then you can transfer in an all other thing ok so, essentially you have to ensure that there are certain protocols with by which we can access multiple systems can access right.

At the same time, but we need some specific support for it. Now you can also attach some of you might have seen SATA disk that is has got USB interface and attached to your laptop or a PC alright so, the fact that certain types of commands are possible on the SATA disk.

For example, something called we will come to this later for parallelism you are something called NCQ-Native Command Queuing. There are multiple requests to the SATA device can be queued up and it can essentially allow you to do multiple things at the same time it can queue things.

Now, even if you have that support does not mean that are the USB level you might have support for it. The SATA disk might have it, but may not be unless it exported at the USB level also that is its in something you have to keep in track so, there is just if you look at the LINUX situation.

(Refer Slide Time: 23:20)



So, basically what is happening is at LINUX kind of system something's are done on a system right, something happened the hardware level, some things is happens at the kernel level. The operating system level you might call it something happens at the user level. Of course, operating system is both user level things and kernel level things, but the kernel thing is a protected subsystem right and you where is the one where things are operating at as a process level right and it you are user is writing this code.

Now, the USB devices here there is a host controller this itself is a device you are has to another driver for the host controller itself and then there is a USB core which is basically the thing which does the protocol for USB devices and then there are specific USB device drivers. For example, you might have the file system layer on top of this or a block layer on top of it could also be a net layer for example, if its a dongle right for example, this kind of things etcetera.

Sometimes there are usbfs also available in some systems. The which you provide all this information whatever has been figured out right through what is call the proc file system. In LINUX there something called a slash proc file system and if you use essentially all that kernel information can made available in a tree like structure the proc file system and anybody who has got that permissions to look at the files can see it and then because you can be writing your own code because you got the information now about the device.

What its characteristics are? Now you can write your own driver using information that has been gleaned out from through the proc file system. Of course, in that still little bit route if it if you are not route in order to anything this way. So, you will be able to because you have special privileges even as person has written user mode driver route permissions are there on it. Therefore, you can use the information that there of the splash proc file system and be able access the functionality of the USB device so, user mode drivers are possible user applications are possible let me user application directly use this part.

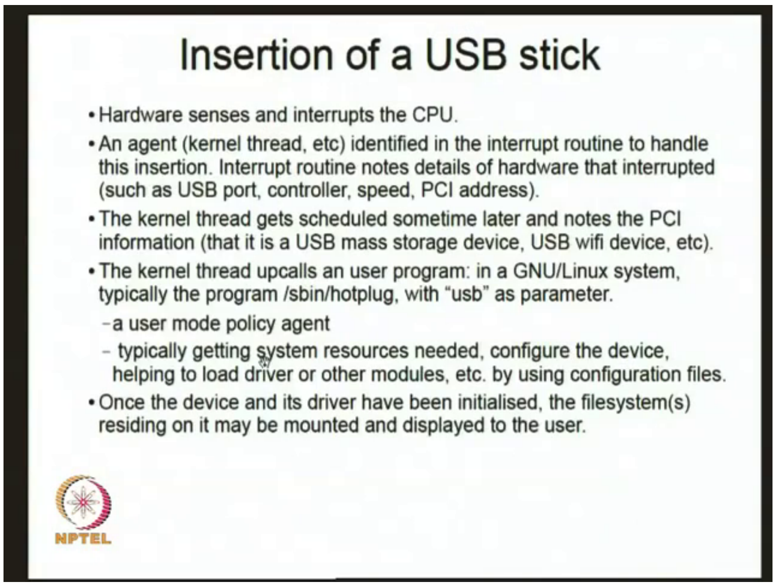
For example, you are let us say your browser for example, is using it wants to upload some files from the USB device so, the file system is already attached mounted therefore, the application using standard f freed and all those things. It accesses it and of course, somebody translating it into file system related calls FAT file system calls. And then that in turn is going to use the device driver and finally, all the way up till USB device where the information to talk ok so, there is USB frame out that is available in this kind of systems in LINUX kind of systems and it is got multiple levels.

I just want to mention that this is typical of any storage system you will find that there is somewhere hardware. It could be a disk it could be a flash device tape whatever it is there is some hardware related things some way to control those things. Some protocol related things because finally, what is happening is that the devices are all the all the time changing right.

You will find that the tape of 1950s it is not the same tape as of today. They will use some different types of technologies and certain things what a feasible there may not be feasible. There something is the technologies all the time changing, but if everything keeps changing all the time then you get into your problem so, what you want to do is to abstract away certain things so, that certain core things do not change and user using that core we are able to talk to even newer things that come out. So, the USB core is basically one of those kind of things is a protocol level protocol specific things so, that you can talk to there is a some specific set of operations that you can depend upon using that you can do you can handle old devices or also new devices ok.

And any specific thing any specific thing, but for example, storage or itself Wi-Fi whatever it is that is handled it specifically through this so, specificity comes from here generality comes from here and the hardware related aspect come from here.

(Refer Slide Time: 28:35)



Insertion of a USB stick

- Hardware senses and interrupts the CPU.
- An agent (kernel thread, etc) identified in the interrupt routine to handle this insertion. Interrupt routine notes details of hardware that interrupted (such as USB port, controller, speed, PCI address).
- The kernel thread gets scheduled sometime later and notes the PCI information (that it is a USB mass storage device, USB wifi device, etc).
- The kernel thread upcalls an user program: in a GNU/Linux system, typically the program /sbin/hotplug, with "usb" as parameter.
  - a user mode policy agent
  - typically getting system resources needed, configure the device, helping to load driver or other modules, etc. by using configuration files.
- Once the device and its driver have been initialised, the filesystem(s) residing on it may be mounted and displayed to the user.

So, let us just try to understand what happens when I insert a USB stick into a system. First of all the hardware senses an interrupts the CPU right because electrically it is sensed and you interrupt the CPU fairly have to identify some agent. In interrupting basically interrupting does some basic things and it identify the kernel thread that is going to be initiated to handle the insertion.

The basically what the interrupting does all it does is it knows which port was there. For example, whether it was this port or this port usually further USB can have a multiple ports right its says came from this side or from this side. So, that has to be identify also the controller who is going to there could be a hub there could be whatever. When they you might have heard about USB hubs right there could be a hub there are to could be multiple hubs also there right it has to be have to figure out who is going to control USB.

The speed on the PCI address once upon a time USB the right USBs where running at lower let us say speeds. Now we have USB 1.0, 2.0, 3.0 somebody has to know what the speed it is. In a sense most of the designs are some extent self identifying the tell you what speed at we said can I operate and also typically these devices has some memory for system memory and that has got all the details about what I/O address is comfortable with, what are the PCI addresses it should be map to etcetera some of the details are here ok.

In a sense the device itself gives information because it knows about itself so, it can tell the kernel which has not. Since, differ device before what are its characteristics so, the interrupt routine only does on simple things first because interrupt routine definition should not take too long so, only simple things are initially captured and then the interrupt controller interrupt routine. Basically, says that this is a kernel thread is going to do some work and this guys going to use informational (Refer Time: 31:00). The kernel thread gets scheduled sometime later and notes the PCI information. That it is a use USB mass storage device whether it is this device that device whatever and then the kernel thread what is do it. It calls an user program typically it is a user program why because there is so, many different things you can do USB is you know you can just imagine how many types of USB device there here.

So, you do not want to put all that function and in the kernel because the kernels already big you do not want to put too many things in it so, for example, there is a program called slash has been hot plug. What is hot plug? The ability to insert things in and out without shutting down the system and restarting it we can put it in analyze system. So, there is similar called hot plug where I can put the USB device hot when the system is running and it basically gives you some control at the user level. Where the main user level again mean at the person using the memory stick. I am talking about the level of the person who has configured a system so, that this kind of thing can easily used so, and I basically this is a the policy is now at the user level not at the kernel level.

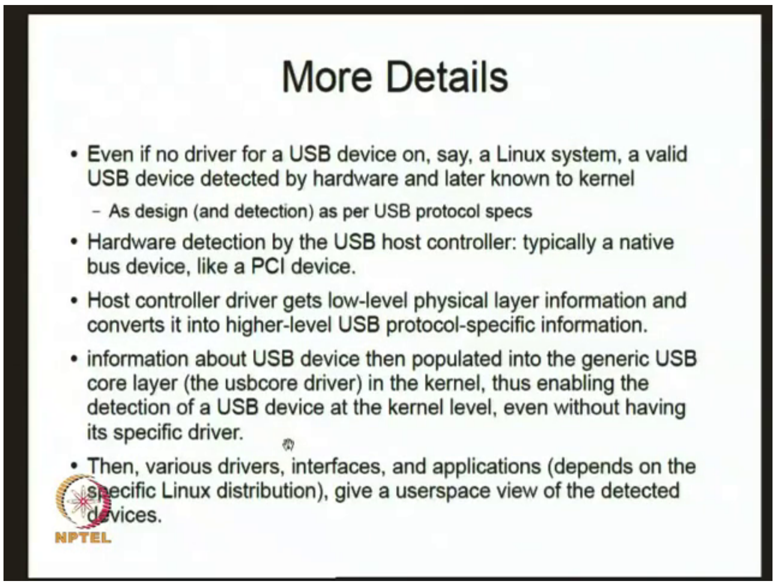
The kernel level means it is true for everybody it is going to be valid for essentially for everybody and user policy is not means that you might want to have your own specific kinds of approaches to how handle the device. For example, it will be that you have an sbin/hotplug and you are in a very sensitive system, no outside devices should allowed. That guy immediately says get lost. I think some of you know that lot of the problems about viruses that are spread usually somebody bringing in outside devices and touching it to in critical installations and often times there is mathematic program that transfer insert a particular device like CD or USB.

And that is basically user level policy in some sense what should happen when I put in and that can be good for most of us, but sometimes it can be bad viruses travel through this automatic route. So, this program like sbin/hotplug what it does is. It notices that the USB device with this with port what kind of controller what kind of a device it is all stuff is, what kind of a driver is going to be has to be used to understand this device all the since with that. So, it figures out what resources are needed it configure the device it helps to load the driver. For example, the mass storage device then you might want to do the it is going to load the driver and using configuration files ok.

For example, more is configure the device it may be that I have to let the device interrupt at a particular interrupt line. If I want to talk to the device I might have to give it some kernel memory so, that it can prepare certain command strings. For example, when we have a SCSI device you need to have something called as SCSI control block. We as used to talk to the device and vice versa there is some if want the device to send back information. It needs to know what kernel unity used because the memory that is being given its not within the device that is going to be use sorry within the device. It is something which it has to pick it up from the host kernel memory ok so, given these things you have to do some configuration.

So, how is the device and driver have been initialized? Then basically you can mount the system and display it to the user. Like for example, when you on a regular system, when you update, when you sorry insert the device you get some kind of graphical user interface. You get right saying what you want to do with it right this kind of things all this things can be done so, let us just look at bit more some more detail.

(Refer Slide Time: 36:09)



More Details

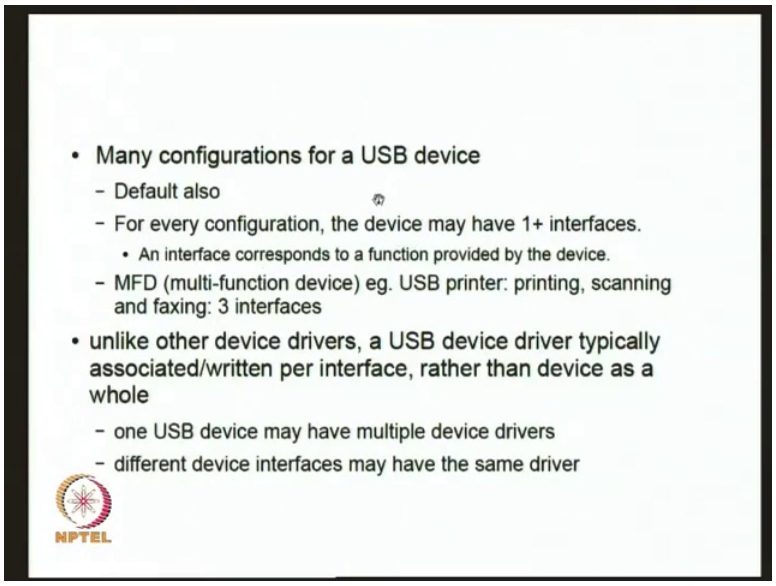
- Even if no driver for a USB device on, say, a Linux system, a valid USB device detected by hardware and later known to kernel
  - As design (and detection) as per USB protocol specs
- Hardware detection by the USB host controller: typically a native bus device, like a PCI device.
- Host controller driver gets low-level physical layer information and converts it into higher-level USB protocol-specific information.
- information about USB device then populated into the generic USB core layer (the usbcore driver) in the kernel, thus enabling the detection of a USB device at the kernel level, even without having its specific driver.
- Then, various drivers, interfaces, and applications (depends on the specific Linux distribution), give a userspace view of the detected devices.

Even if there is no drivers for USB device you can still take the USB device because this is part of the USB protocol specs. That the driver actually does the specific aspect of the device whether its mass storage or its wireless or whatever it is there is a camera all those things. That part of it is done by the device specific part of it, but up till that generic part it is done by the kernel because if it is LINUX kernel which has support for USB devices. The generic part will always be there; that means, that sorry this part is always be there because if always there it can get to the point where it can do whatever generically. It can do this is required for specificity so, hardware detection that used USB host controller

And basically, the host controller driver gets low level physical information and converts it into higher level USB protocol specific information and this information is populated into generic USB core layer this USB core driver so, in a sense the kernel knows about the USB device. Now if we are not now exactly how to use it that is the different story, but it knows what devices were inserted. Then depending on the availability of other drivers interface and applications depends on specific identification. It gives user space of view of that detected devices. Now, this is what is the thing that gives you the thing which is user friendly point of view this point. The kernel knows about it, but beyond that kernel anything because it does not know what to do with it.

You need to gave additional piece of information the specific driver for accessing the particular USB device. In other case mass storage you need to have a USB mass storage device driver and USB mass storage device driver assumes that is the SCSI device and it knows that it has to send and receive. I send SCSI commands and it gets back responses.

(Refer Slide Time: 38:40)

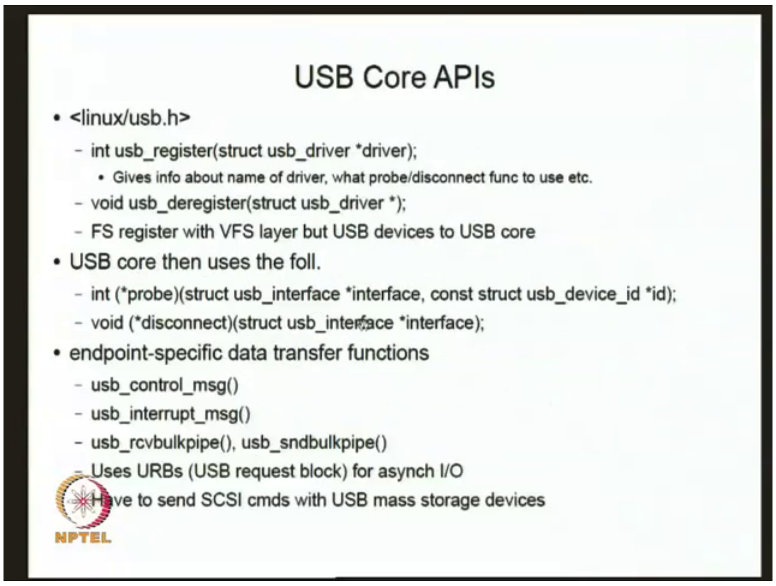


- Many configurations for a USB device
  - Default also
  - For every configuration, the device may have 1+ interfaces.
    - An interface corresponds to a function provided by the device.
  - MFD (multi-function device) eg. USB printer: printing, scanning and faxing: 3 interfaces
- unlike other device drivers, a USB device driver typically associated/written per interface, rather than device as a whole
  - one USB device may have multiple device drivers
  - different device interfaces may have the same driver

So, again the lot of interesting things about this thing basically there are multiple configurations for a USB device so, I can have multiple interfaces and already talked about this so, for example. I have multifunction devices USB printer for example, this what printing scanning and faxing so, that is why there are multiple device drivers. For example, like camera will have sorry the printer will have multiple device driver. It is not that one device drivers actually handles all three functionalities there will be a device driver for each functionality in typically in these kind of situations USB printer kind of situations ok.

So, something interesting over USB devices that one USB device may have multiple device drivers and different device interfaces may have the same driver also that also its possible both these are not possible so, we will just quickly look at some aspect relating to how LINUX actually handles it.

(Refer Slide Time: 39:35)



USB Core APIs

- <linux/usb.h>
  - int usb_register(struct usb_driver *driver);
    - Gives info about name of driver, what probe/disconnect func to use etc.
  - void usb_deregister(struct usb_driver *);
  - FS register with VFS layer but USB devices to USB core
- USB core then uses the foll.
  - int (*probe)(struct usb_interface *interface, const struct usb_device_id *id);
  - void (*disconnect)(struct usb_interface *interface);
- endpoint-specific data transfer functions
  - usb_control_msg()
  - usb_interrupt_msg()
  - usb_rcvbulkpipe(), usb_sndbulkpipe()
  - Uses URBs (USB request block) for asynch I/O
  - Have to send SCSI cmds with USB mass storage devices

I think if you look at the LINUX kernel, you will find that there are some header fills some a particular place and that so, this is the notation used usually means have a slash user enclose etcetera. If you looking there you have this functional it usb_resister and usb_deregister so, what is this usb_register doing? It is gives information about name of driver, what probe/disconnect function to use so, basically what we are saying is there is a you want to say basically what is happening is that I want to I have detect the device.

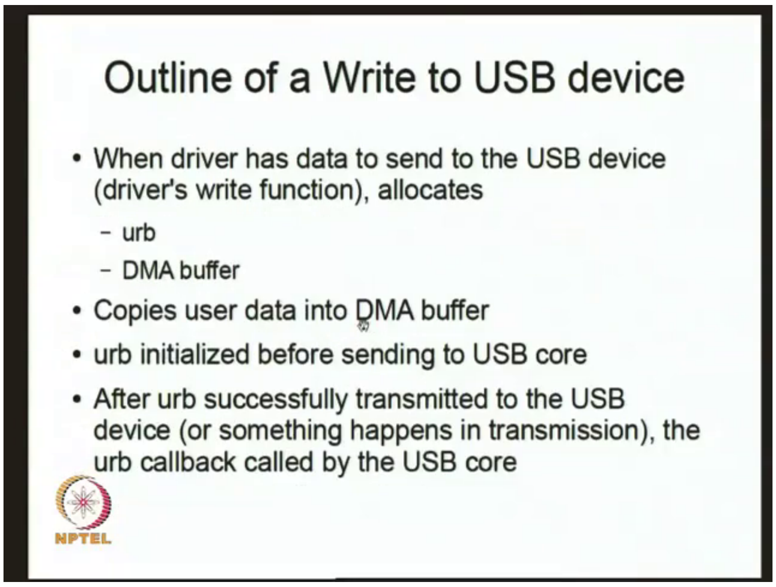
Once that devices in detected you have to someone gives you additional information about which driver is going to act on it use it. For example, all that has happened is I had to say which driver is going to be used because I am at this point here. I want to say what driver is going to be use on it so, that is what is going on at this point so, what I am going to do is. I am going to say now that you know that I this device exists, please use this particular let us say function. This pro function to know that everything is that this is actually the function that has to be used for accessing this device and the (Refer Time: 41:27) it is by as a name your driver also I give it so, once I am done with the use of this particular device or can do usb_deregister.

So, normally you tells out that there is a similarity might see you see that file systems usually register something called the VFS layer. Here the USB the devices due to USB core. Now that the USB core has been told not this is the driver name, the driver itself has some functionality one of them is program disconnect so, the USB core. Then uses the name of the driver in the driver it has got two let us say I said two functions program disconnect of course, it is the other functions. Then USB core calls a probe function and convinces also these are right function right functionality.

That is somebody is giving information about how I should be accessed the kernel in turn (Refer Time: 42:30) again you just the way of checking and then you can do some specific data transfer functions so, you can send control messages, you can send interrupt messages. For example, I can cancel the request suppose I am doing something's. I want to cancel it its possible in some cases to say please stop that copies that is going on that I control that is interrupt message and these are basically the things which tell you how to use how to send bulk in bulk data this one's ok.

The way you should do all these things is basically you have something called USB request block and because the devices are slow it uses asynchronous I/O and basically what happens is you send SCSI commands to the USB mass storage devices the exact specifics are as follows. So, first of all what happens is that when the driver has data to send to USB device you have to allocate what is called urb.

(Refer Slide Time: 43:48)



Outline of a Write to USB device

- When driver has data to send to the USB device (driver's write function), allocates
  - urb
  - DMA buffer
- Copies user data into DMA buffer
- urb initialized before sending to USB core
- After urb successfully transmitted to the USB device (or something happens in transmission), the urb callback called by the USB core

What is urb? urb again is the USB request block ok.

Some piece of kernel memory that is allocated and you also allocate a DMA buffer because what is the DMA buffer. The DMA buffer is basically web programming the hardware system so, that they can transfer data efficiently so, there is if there will have it will send to have something like count also sector so, that keep strike of how many bytes have been transfer always kind of things so, you also want to shaded DMA buffer where from your actual transferring things and also I want to program the DMA engine itself. So, the data that has to be sent for example, if we are talking the write you copy the user data into DMA buffer.

The user data could be coming from user space and then it has to be copied onto DMA buffer, but means that there can these is a kernel function. This actually has to be careful that user is not giving you something which can create from security problems. It has to be mapped it has to be a legitimate pieces of in the it is not an illegal address for example, all those things are checked by the kernel and then scrub into DMA buffer ok.

Now, once you have the allocation from urb and you have the DMA buffer where the data that has to be transferred is already copied into. Then you can initialize what has been the urb allocation you are look you initialize it and then you send it to USB core. The urb when its initialized you also send it what kind of functions to use for doing any by transfers is possible and then also it gives you a callback that is ones a function is completed who should be called. So, now once it is done the urb is transmitted to the USB device now this can succeed this can also fail whatever reasons.

Instead of whatever happens the urb is called is callback is done so, that to complete whatever either it because it completed successfully completed it. Therefore, you might have to do some cleanup what if something that happens also you might want to be clean up that also. So, in a sense what is happening is that if you think about this if you look at this particular framework right. What is going on here? You want to send some you want to write to something is application out here is wants to right something. Once again with the regarding user buffer that is above for has to be copied into kernel first and that is going to be done by the file system layer or you can be doing it directly from user mode drivers. You have user mode driver you will not got the file system layer you will not actually directly come through this ok so, that is even my we are talking about copying its from user space to kernel space.

Normally, it will be done with the file system because normally what is the situation I have a FAT file system FAT32 files system and user level actually uses the C library and

C library as something called F read and the F write. F write actually it takes it from your user buffer and copies it into the buffer that is used by the library. That in turn is copied into the file system it also copies it into a kernel memory there and then in the USB device driver now know that is already in the kernel space may not need actually copied into another place. It may be necessary in case that DMA buffers need to be in a particular place. Sometimes you will find that in LINUX systems, the DMA it cannot the DMA can only happen if the in the layer 24 bit physical address space the memory.

There are some restrictions of that kind. So, then you might want to copy it into the lower 32 bit sorry lower 24 bit physical address space. Those kind of this all done by various subsystems total. Finally, it is in the DMA buffer and then you are now asking it to you create a urb now because now we are somewhere around this point. You create a request buffer which has all the data and you are asking it to use a specific type of command. For example, the I mention the send block right sndbulkpipe sorry use this particular function saying that use this as a way to transfer information and this might because I am using way of this might actually do the asynchronous I/O part of it.

And then the host controller there might be some additional things. For example, it may be that whatever information I have to send has to be chopped up into pieces and sent. It may be that this guy can understand only some different units. Now is that for example, if you take your regular USB memory stick it have its it has its own notion of what is called read size, write size and erase size so, if you ask it to right a small portion alright it could has to reuse that part.

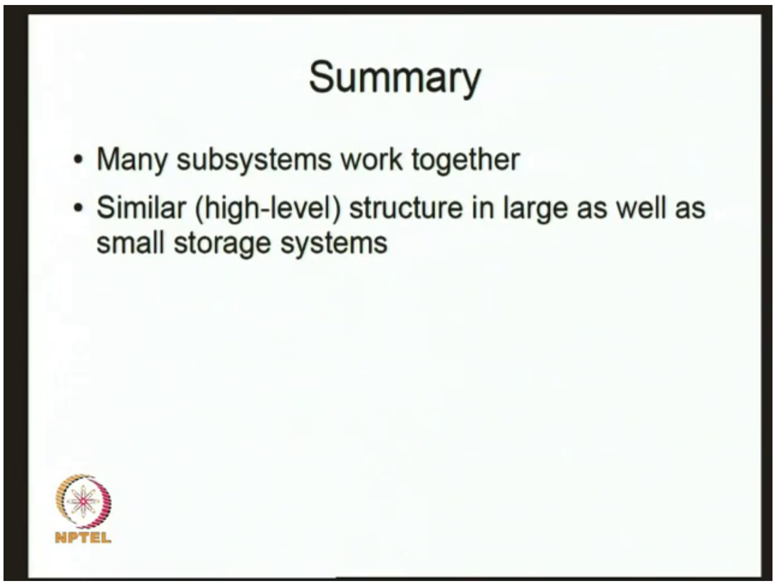
Somebody has to there was a whole thing then write that part right. So, it may be that some of those kind of things are happening at this level. It could be that you are writing in terms of units of what is the size of fast file system block size. It can be 4 kilobytes, it could be I think with different as a FAT systems you have different block sizes alright. The thruster some various things are different right so, because there are difference the unit at which you are writing is different for it.

For different file systems that may not be the same as what is there in the USB device so, somebody has to do what is what we call that you last time segmentation and reassembly and that is happening at this point and once that happens the data is sent out for writing and once it is completed. The host controller USB host controller will get a notification on the device saying it is complete. It will some then interrupt the USB host controller actually is notified. It typically host controller will handle all the interrupt for some time till it decides to interrupt the CPU ok.

Once it has to interrupt the CPU then you know that the operation that you sent out earliest complete and then in turn what happens is that that callback function that was registered right will be called so, that the any cleanup activities whatever has to be done can be initiated so, this roughly the model of how transfers take place and may not really discuss our things. Like how concurrent operations can take place, but there are some details that also have to be further looked into ok so, what is interesting for us?

Is basically how many things are involved before you can get USB to work. You have an user application which actually shows you the GUI, saying what do you want to do with this thing. They want to think of it as a mass storage device or do you want to use it as a some over type of device right like I mentioned of android device right. if this is an option. All of those things are user level applications and then there is a FAT32 file system typically for USB devices, but what applications are using to write and there are other pieces of things are already there which have to all work together before we can get any data in and out to the system ok.

(Refer Slide Time: 53:32)



Summary

- Many subsystems work together
- Similar (high-level) structure in large as well as small storage systems

So, basically many subsystems have to work together and you will find that this kind of structure is there in where it look at small systems or big systems. For example, suppose you take a very big huge massive storage system right. First of all, I need to know where to write to right because I need to know that such thing exists first of all. Someone has tell me that this device exists, it could be artificially created one. That is it is actually composed of 10 1 terabyte disk and somebody is making it look as you said 10 terabyte disk and I need to know that somebody giving me that that there is virtual 10 terabyte device every day so, some kind of discovery. Instead of some need to know you can have both the devices whatever name I do it to it so, that I can start incorporating it into my system so, that I can say that this exists now I can start using it so, there is some aspect of device discovery there is an aspect about how you can take the information there which has to be recognize has a file system.

For example, somewhere has to mount it once it is mounted then the user know user application has know where is mounted so, and then once it is mounted. Then you have some notions about at in what units of granularity, you want to some information and that might be different from what these devices are comfortable with. Somebody has to do some translations all the sense are required. You will find then this is true for almost any logical system, you will find that there are a lot of multiple aspects are to be handled different interfaces will be there which will be different layers will be there which will handle different things.

You will not really talked about things like encryption decryption etcetera. It could be possible that you want to encrypt things yourself because nowadays you have so many USB so, you can leave it here and there and forget it. And somebody can look at some sensitive information so, you might want to have encryption somewhere. Now where should the encryption be should be at user level or should be at this level. Ideally, it should be somewhere at this level so, that but then there is I guess as talked to last time that is device the key management information. So, there are lots of issue all these kind I have to be all looked into and so, a system perspective is important. We will continue with next time on some other examples right.

Thanks.