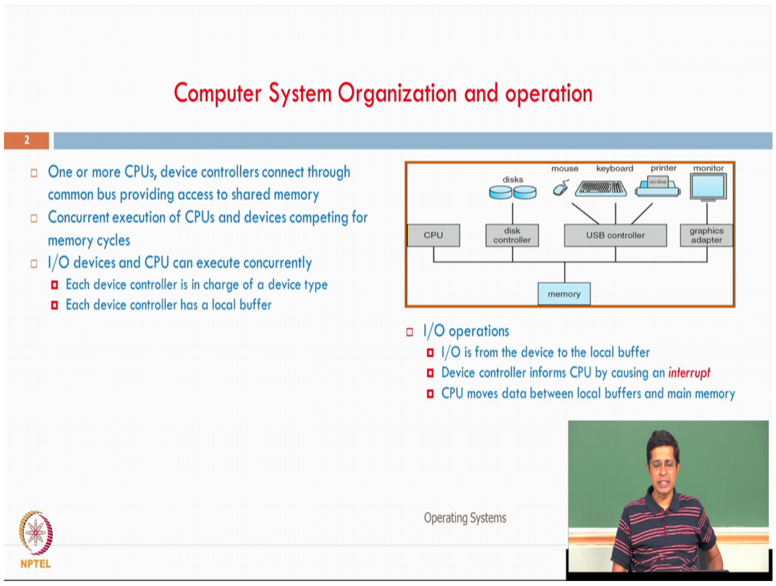
Information Security-3

Prof. V Kamakoti
Department of Computer science and Engineering
Indian Institute of Technology Madras

Basics of Unix and Network Administration
Operating System Introduction
Mod01 Lecture 02
Module 2: Storage Hierarchy, Exceptions, Interrupts and Traps

So welcome to module 2. In this module, we will be talking about storage hierarchy, exceptions, interrupts and traps. These are concepts that are extremely important for one to understand, the functionalities of the operating system (operating) something we expect from the operating system, we want it to do certain management's, management of process, management of files, management of memory etcetera. For to understand these functionalities in good depth, we need to understand certain terminologies, which are very important namely we need to understand what do you mean by storage hierarchy? What do you mean by exceptions? Exceptions is of 2 types, interrupts and traps. So in module 2, we will cover these things in detail.

(Refer Slide Time: 1:03)


Computer System Organization and operation

2

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles
- I/O devices and CPU can execute concurrently
  - Each device controller is in charge of a device type
  - Each device controller has a local buffer

- I/O operations
  - I/O is from the device to the local buffer
  - Device controller informs CPU by causing an *interrupt*
  - CPU moves data between local buffers and main memory

Operating Systems

NPTEL

When you look at traditional computer system organization. So you basically have a bus which is given by the lines there and then you have a memory, then you have a CPU and then you have a set of peripherals, on the screen, you see five peripherals, namely the discs, the mouse, the keyboard, printer and then a monitor. Each of these peripherals have certain intrinsic value in terms of suppose we take a disc, there is a way a disc will store certain data. There is a way the disc will allow access to the data. So different disc could have different formats. Similarly, different printers could have different formats by which they will do the printing, they could have different functionalities.

Now between the disc and the bus as you see there in the slide, there is a disc controller. The disc controller understands the language of the CPU and also of the underlying peripheral namely the discussion. So the disc controller can be termed as an adapter, which adapts the functionalities required by CPU to be carried out on the disc. It basically translates the need of the CPU to a language understandable by the peripheral it controls. So for the disc we have disc controller. Similarly, for the monitor we have a graphics controller or graphics adapter and for the mouse other IO devices we also have this USB controller.

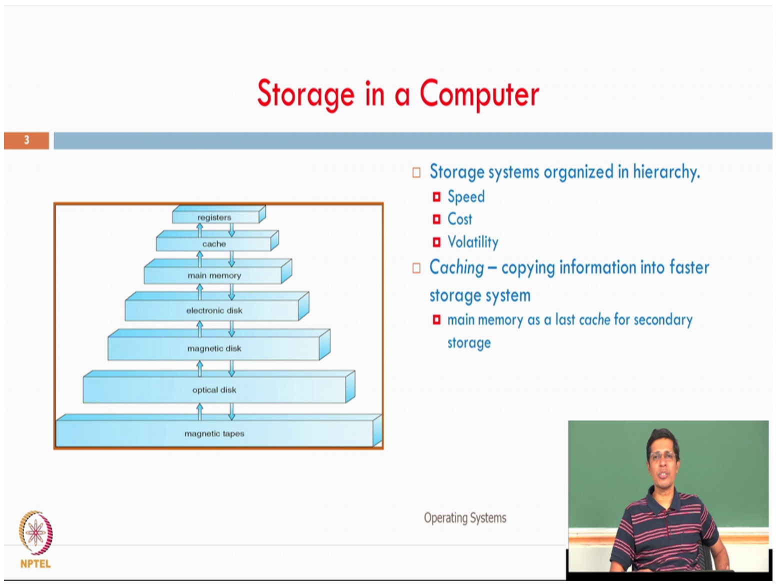
So the CPU can communicate to this the peripherals through the corresponding controllers and note that these controllers are basically hardware. They set between the bus and the peripheral. Now, these CPUs they actually work concurrently with the devices. So when a disc is actually performing certain activities, the CPU can also start doing some activities and both of them colatch when they need the bus. So the bus itself is a sheared resource, which needs to be scheduled, term scheduling means that a particular sheared resource is allocated to one of the entities, which request for using the sheared resource at a particular point of time.

So the bus itself is a sheared resource and that needs to be scheduled properly and the IO devices basically need to talk to the CPU or the IO devices also need to talk to the memory. So there are different ways by which an IO device can communicate. The data from an IO device can be moved back and forth from the system, for example, if the CPU wants to directly read from IO device, each of these controllers could have a local storage. So the IO device can basically write into this local storage and then give an interrupt. An interrupt is nothing but a signal to the CPU that I have some data available to you why do not you take and use it? So who will inform the CPU? The device controller will inform the CPU if for example, the keyboard wants to talk to the CPU then through the USB controller, it can basically send the interrupt to the CPU saying that it has some data which the CPU can use.

Now, the CPU will be responsible for taking that data from the device controller and basically using it. There is another way by which data can be transferred back and forth from the system and that is through the memory. So later in the memory management we will look at something called DMA which is called direct memory access. So when a peripheral wants to send data into the system, it would for example, if I want to transfer a file you will transfer the content of the file from the disc to the memory without involving the CPU. So the term direct memory access, what do you mean by direct? The term direct essentially means that I am bypassing the CPU. So an IO operation can also happen between the peripheral and the memory without the CPU involved and this particular operation is called direct memory access.

So to some up what we have seen, the organization of the typical computer system would be as a bus, on the bus the CPU will be attach to, the memory will be attached to and there will be (())(5:39) peripherals and each of these peripherals will have a controller, which is hardware. Now, the CPU talks to these peripherals and vice versa through this hardware controller and this is how a typical computer system is organized and the IO operations takes place.

(Refer Slide Time: 5:58)



Storage in a Computer

3

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility
- *Caching* – copying information into faster storage system
  - main memory as a last *cache* for secondary storage

Now, coming to the storage, storage is actually one of the major innovations in computing. This allows you to write something and remember something, right. So the time that I could remember actually is something very very important, for example, if I store whatever I want to remember inside my CPU then I could access it very very fast. If I store it outside the CPU, the time require to basically retrieve what I have stored will take little more time. So it will be higher and then I want to permanently store something then I store it in a disc. What do you mean by permanent? In the sense that I want to store it till I want to delete it.

So even if I switch off the system, the data will be stored and I could retrieve whenever I switch on the system. So this is basically you called volatility, right. So if I store it in inside the CPU or in the main memory, which we call as the random access memory. When I switch off the power, the data gets lost. So these are all volatile storage, but if I go and store it in a disc or something more there as you see in the slide if I have to store it on a magnetic disc or optical disc or magnetic tapes, after I switch off the power still whatever I have saved will be there and I could retrieve at any later point of time. So those storages are called nonvolatile storage.

So as we see we have registers, we have cache memory, we have main memory then we have electronic disc, then magnetic disc, optical disc and magnetic tapes. The last four namely the electronic disc, magnetic disc, optical disc and magnetic tapes, they are non-volatile storage while main memory, cache and registers are all volatile storage.

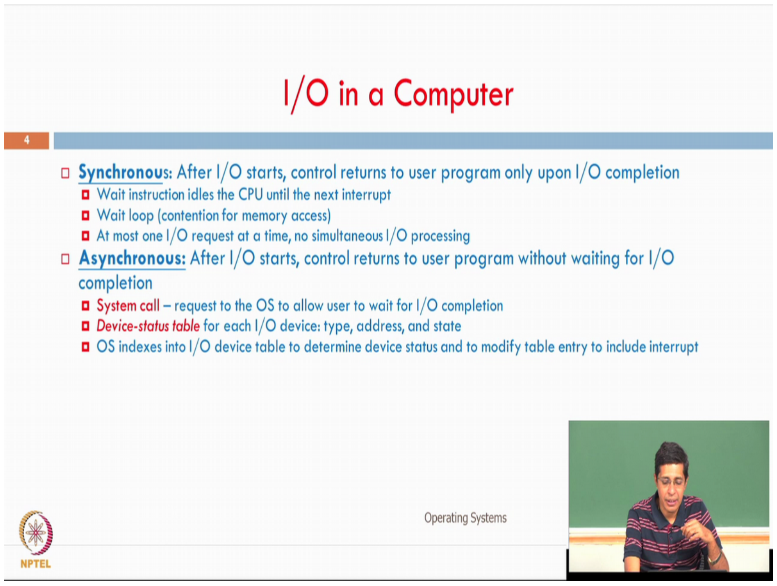
Now, as I go down the hierarchy from registers to magnetic tapes, the amount of time I spend to basically store a data and retrieve it back, to read and write from that memory, this time actually increases. The time I required to read and write from a register would be very fast, it will be nanoseconds and also, in cache it will be say, tens of nanoseconds. Main memory could go more than the Nano scale, but when I go to electronic disc or magnetic disc, optical disc, magnetic tapes, this could be several orders of magnitude slower (())(8:51).

So we are looking at volatility and going towards non-volatility as we go from top to bottom and again the speed of access basically, access means read or write from a particular storage entity, the time for access actually increases in a very fast ways (())(9:10) as we go from registers to magnetic tapes, but on the other hand, the cost per bit what is the cost I incur (()) (9:19) to create one bit on a register, to create one bit on a cache, to create one bit on a memory and so on so for the cost also decreases.

So this is how the storage hierarchy is built in a computing system and one of the important thing that we should note here is that at every level, I would have something called a cache memory. The cache memory is that traditionally we see is the cache memory that is inside a chip or inside a CPU, but even for file systems, there something called buffer cache, even when you download through the web your browser has a browser cache. What do you mean by this cache memory is that I am retrieving say, from a electronic disc on to the main memory and I am writing it back, I am retrieving something which I am using it quite frequently. So instead of every time going to the electronic disc, I will take some amount of data and keep it in the main memory and reuse that in the main memory and at a later point of time store it back into the electronic disc.

So every time I need not go to the electronic disc, I will create some locations in the main memory and keep using it by this, every time I am want to access a data, I access it from main memory rather than the electronic disc. By this my speed of access actually basically increases here. So at any level a cache is something which will take the data from the level below it and keep it for reuse and at a later stage we will send it back to the lower level. So this is the notion of the cache. So as we go through the operating systems and networking, we will find lot of places where cache memory, caching as a concept is being employed basically to improve access times and that is also very important for us to understand in the context of storage.

(Refer Slide Time: 11:20)



I/O in a Computer

4

- **Synchronous:** After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request at a time, no simultaneous I/O processing
- **Asynchronous:** After I/O starts, control returns to user program without waiting for I/O completion
  - System call – request to the OS to allow user to wait for I/O completion
  - *Device-status table* for each I/O device: type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Operating Systems

NPTEL

The next important thing is the IO in a computer. So everything is basically IO, today I use a mobile phone, I use a laptop, I use a projector. Now you are viewing this course over a web, it is this all input output operations. So we need to understand how IO works in a computer and please, also note that all we are talking from a security point of view, all the access that comes all the sort (())(11:45) of vulnerabilities that we see in practice, also basically depends upon the IO, the strength of the IO, the security strength of the IO mechanism.

So for us to appreciate IO on its (())(11:58), let us understand some of the important aspects of input output operations. The first thing is a synchronous IO. What you mean by a synchronous IO? I am a program in execution and I want to do an IO operation, I start the IO operation and I do not proceed till (I) that IO operation is completed. So I actually wait, I ideally (())(12:23) in the CPU till that IO operation gets completes. So this is basically called a synchronous IO, by doing this synchronous IO at most one IO operation can be initiated and after it completes a next IO operation can happen.

In the case of an asynchronous IO, the program in execution, which we called as a process. The process that I actually wants to do this IO operation, basically it starts this IO operation, it comes back and it starts doing other useful work at some later point of time, there would be a interrupt from that device saying that it has completed. Then it will go and check whether it has completed and it will take some necessary action depending on whether the completion was successful or unsuccessful. This is basically called as asynchronous IO, by doing this asynchronous IO, a process can initiate an IO operation and when that IO operation is basically happening then it can do other useful work, by this we through asynchronous IO operation, the computation and the communication can be sort of a done in a concurrent fashion.

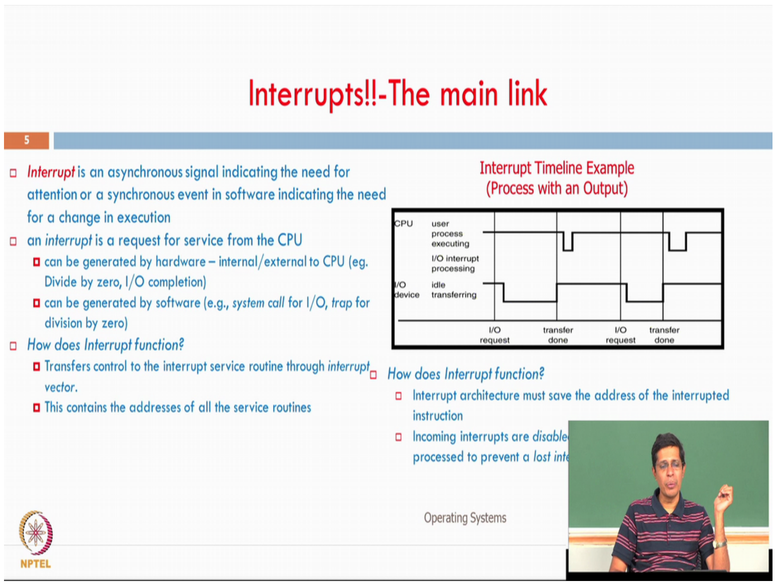
So asynchronous operation crucially depends upon certain things like a system call. What is a system call? It is a request to OS to start an operation and sometimes, it is the request to an OS to make the process wait till an IO is completed. So the typical way an asynchronous operation would happen is, there are typical system, there would be more than one processes that are executing at a point of time. So there will be process 1 that is executing, there will be another process 2 that also wants to execute.

Now, when process 1 starts executing, it will execute on the CPU then it will come to a stage where it needs to do an IO operation, then immediately it tells, I want to do an IO operation and till the IO operation is completed please suspend (())(14:22) me. So the process is actually blocked and the IO operation will start happening and when the IO operation is happening, the CPU is not involve in that IO operation. So the (CU) CPU can start working on the second process, which is waiting for it and once, the IO operation is completed then an interrupt comes it to the CPU saying IO is completed then the process 1, which is waiting for the IO to complete can start working again. So this is also a different notion of a asynchronous communication where (())(14:57) IO operation and a computation can happen concurrently.

Now system call is a way by which a process, which is nothing but a program in execution will request the OS for the certain services and the typical operating system will also maintain something called a device status table, which maintained for each IO device and which will contain information or attributes of about that device, its type, the addresses to which, it is mapped and state and this is organized as a table. So depending on the device every device will be given an id or an index and the OS will actually index into the table using a device id and go and find about what is happening with the device. So this is how a IO in a computer is a basically managed.

So we need to understand what is a system call, we need to understand what are device status table and so even from a security pointy of view, understanding of these concepts are very important, because tomorrow a attack actually happens when we look at the secure aspects through these system calls, through certain loop holds (())(16:11) in these system calls and by manipulation of some of these sheared tables including device status tables. So these are some of the things that we need to keep in mind and understand.

(Refer Slide Time: 16:25)



Interrupts!!-The main link

5

- *Interrupt* is an asynchronous signal indicating the need for attention or a synchronous event in software indicating the need for a change in execution
- an *interrupt* is a request for service from the CPU
  - can be generated by hardware – internal/external to CPU (eg. Divide by zero, I/O completion)
  - can be generated by software (e.g., *system call* for I/O, *trap* for division by zero)
- *How does Interrupt function?*
  - Transfers control to the interrupt service routine through *interrupt* vector.
  - This contains the addresses of all the service routines

Interrupt Timeline Example
(Process with an Output)

- *How does Interrupt function?*
  - Interrupt architecture must save the address of the interrupted instruction
  - Incoming interrupts are *disable*
    processed to prevent a *lost inte*

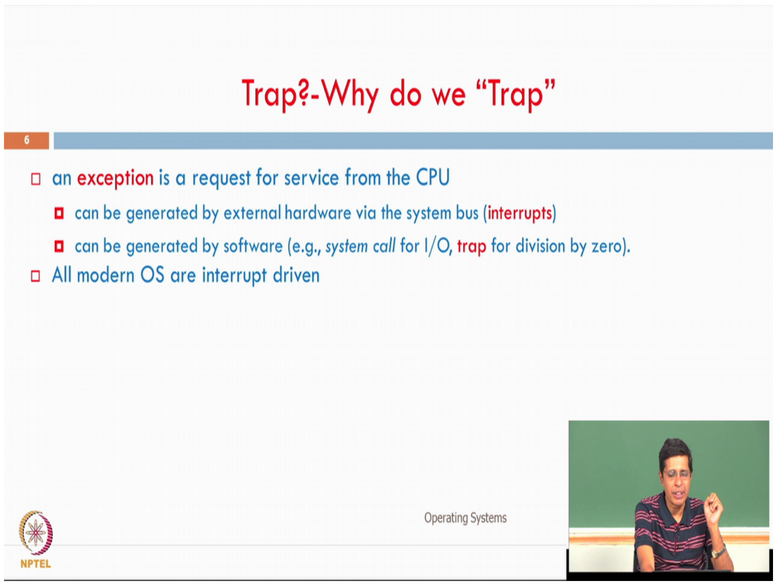
Operating Systems

NPTEL

Now we will understand very importantly what we mean by an interrupt. All the operating system today are interrupt driven, suppose you take a mobile phone, the mobile phone is dormant, when you press a key then the screen comes up, when you touch the screen then something happen. So the mobile phone is actually an example of a reactive operating system. It reacts or it reacts to an event so sometime, it is also called event driven operating system. Sometime it is also called an interrupt driven operating system. Almost, all the OS that you see today in practice are interrupt driven operating system, because when I go and touch the touch screen please, note that there is an interrupt that is generated and then certain actions are being done.

Now what is an interrupt? Interrupt is actually an asynchronous signal indicating the need for attention. An interrupt can also be a synchronous event in software indicating the need for a change in execution. So when we do some aspects of process management, we will understand about asynchronous event, which we call as interrupt and synchronous even which we call as interrupt.

(Refer Slide Time: 17:45)



Trap?-Why do we "Trap"

- an exception is a request for service from the CPU
  - can be generated by external hardware via the system bus (interrupts)
  - can be generated by software (e.g., *system call* for I/O, trap for division by zero).
- All modern OS are interrupt driven

Now in the next slide we will see a little more, I would like to go to the next slide before coming back to this again, different books have different terminologies, but for the purpose of this course we will understand interrupt as follows. There is something called an exception. What is an exception? An exception is a request for some service from the CPU, right. So who is generating this request? The CPU is generating this request.

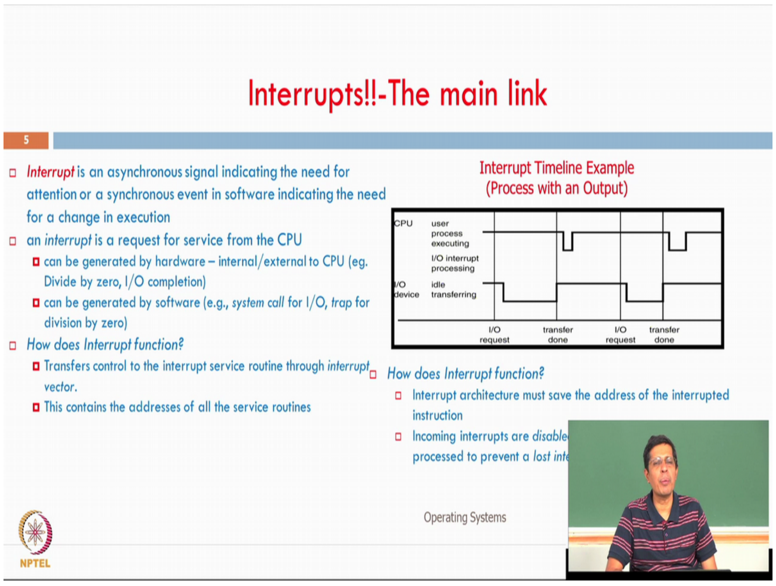
Now what do you mean by CPU generating this request the CPU has found certain exceptional seen arrow. What is an exceptional seen arrow? Exceptions are of 2 types, one is called Trap another is called interrupt right. So many books have different terminologies for this, but for the purpose of this course, this is the terminology I will be following, there something called exception. Exception is something like an exceptional seen arrow that is the CPU has encountered and it is requesting for some attention from the operating system. This exception is of 2 types. Type 1 is called interrupt, type 2 is called trap.

Trap is something that I as a program in execution have created an exceptional seen arrow, for example, divide by zero whose mistake it is? It is the programs mistake. The process that is in execution, it has done a divide by zero. It is trying to do a divide by zero and hence, exceptional seen arrow has come up. Segmentation fault, if you look at my information security 2 course, architectural layer to information security we have talked elaborately about segmentation fault, we have talked about interrupts, exceptions also there, in very great detail right.

So I try to access a segment, I as a process program in execution try to access a part of the memory, which is not intended or allocated to me. This is called a segmentation fault. So who is responsible for it? The program that is in execution is responsible for it. So those we called as traps, suppose a disc has finished operation or in the keyboard, I go and press control C or in the keyboard (I) I am just going and changing the slide, for example like this.

Now these are all exceptional seen arrows that are coming from outside the system outside the CPU created by the IO peripherals or the external hardware. So any request for attention or what we called as an exception comes from outside the CPU that we call it as interrupts. So exceptional seen arrow is a seen arrow where there is a request by the CPU for some special attention by the operating system. This exceptional seen arrow can be of 2 types, one is called interrupt, another is called trap. Trap is something that the program in execution within the CPU has generated an exceptional seen arrow for which an attention is requested for. An interrupt is something external to the CPU chip, which is coming from external peripheral devices etcetera that you need to handle, right. So all the modern OS as I mentioned are actually interrupt driven.

(Refer Slide Time: 20:54)



Interrupts!!-The main link

5

- *Interrupt* is an asynchronous signal indicating the need for attention or a synchronous event in software indicating the need for a change in execution
- an *interrupt* is a request for service from the CPU
  - can be generated by hardware – internal/external to CPU (eg. Divide by zero, I/O completion)
  - can be generated by software (e.g., *system call* for I/O, *trap* for division by zero)
- *How does Interrupt function?*
  - Transfers control to the interrupt service routine through *interrupt vector.*
  - This contains the addresses of all the service routines

Interrupt Timeline Example
(Process with an Output)

- *How does Interrupt function?*
  - Interrupt architecture must save the address of the interrupted instruction
  - Incoming interrupts are *disable*
    processed to prevent a *lost int*

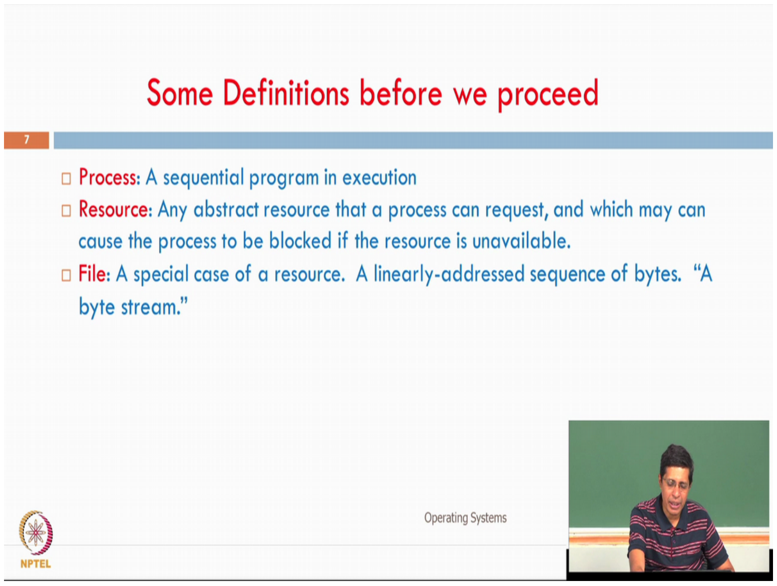
Operating Systems

NPTEL

Now let us go and see how these exceptions are basically handled, right. Now, what happen when there is an exception every exception whether it is a trap or an interrupt as a number? So when there is a number that is given to this. Now there is a table called an exception table which we call as an interrupt table, for every exception there will be a pointer to a code, which needs to be executed for exception, for example, if I do a divide by zero then I go to the interrupt table will tell for example, the divide by zero will be allotted a number say, 10 then we go into the interrupt table and in the 10 th location in that interrupt table we will basically have a pointer or an address of a particular code. If you go to that address then and start executing the code starting at that address then, it will be the service for the divide by zero, basically it will print on your screen divide by zero error and it will terminate.

So for every interrupt, there is a number and when that interrupt is generated, the hardware automatically goes to the interrupt table and index into that interrupt table with that number and in that entry, there is going to be a pointer to a code segment or a code and that code will start executing and that code will do the necessary service that code is call the interrupt service routine. So a very detailed way of how the Intel architecture handles interrupts and exceptions, traps and interrupts namely the exceptions, this has been covered in the information security 2 course, I it request the audience to look into that course that module in that course, it is available on the NPTEL site to get more on this. Actually, the second course we have even written an interrupt service routine from scratch (())(23:02).

(Refer Slide Time: 23:02)


Some Definitions before we proceed

7

- Process: A sequential program in execution
- Resource: Any abstract resource that a process can request, and which may can cause the process to be blocked if the resource is unavailable.
- File: A special case of a resource. A linearly-addressed sequence of bytes. "A byte stream."

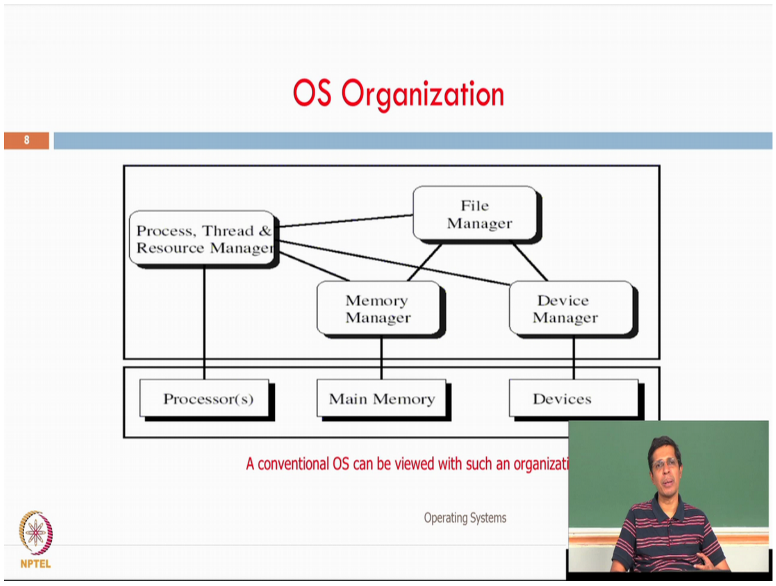
Operating Systems

NPTEL

So before we proceed let us look at these definitions, I have been repeating these definitions. What is a process? It is a sequential program in execution. What is a resource? It is something that a process can request, process can request for a CPU time, yes it is (())(23:18) CPU time is a resource. The process can request for a peripheral. So hence it is requesting for a bus, bus is a resource, process can request for some memory location by saying, I want to read or write into a particular memory location.

The memory location is a resource and when that resource is not available, for example, my CPU is not available or my peripheral is not available, the process can be basically blocked. So if I am blocked blocked means I will cannot execute further till, because I need a resource and that resource is not available, the operating system can go and block me and then allow me to proceed once that resource becomes available. File is actually a special case of a resource. It is actually a very broad definition of a file is a linearly-addressed sequence of bytes or what you call as a byte stream, right. So we will be looking into files also in great detail.

(Refer Slide Time: 24:10)

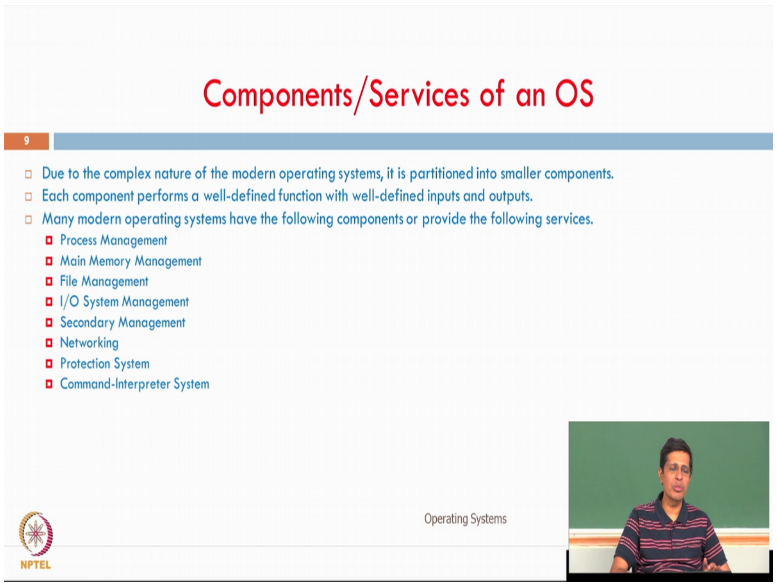


So to sum up if I look at OS, the organization of an operating system, which is very important for us to later appreciate, the secure aspects of operating system design is that. There are 2 layers, 1 layer that you see in the bottom as you see on the slide is the real hardware, I have processors, I today have multicore environment (())(24:30), I could have multiple processes there, then there is a main memory and then there are devices and different modules in the operating system governs each one of them, for example, if we look at a traditional operating system , conventional operating system, even modern operating system.

There is a module which is called process, thread and resource management module manager and that is responsible for handling these compute part namely the processors that is responsible for what you call as COU handling or CPU scheduling, handling etcetera. Then there is a memory manager, which is take which basically handles the main memory, the RAM and there are device managers, which basically as device drivers etcetera, which manages the devices and all of them basically uses the files. What do I work on? I work on a files please, note that If I want to store any information in the computer I can store it only in the form of a file.

So a file is the basic unit of storage in the operating system. So the memory manager, the device manger and the process, thread and resource manager, they basically talk to the file manager to get lot of information from the files. So the memory manager also talks a file managers, the device manager also talks to the file manager, because lot of file related activities happen at the memory, I just mentioned about something called buffers cache that is maintained in the memory, for example and lot of things about files also happen in the device level, because when I want to read a file, I just say read that file then which part of the file from where to read, where it will be located everything basically there is the happens. The entire read operation happens in a very synchronized fashion where the file manager and the device manager are deeply involved.

(Refer Slide Time: 26:22)



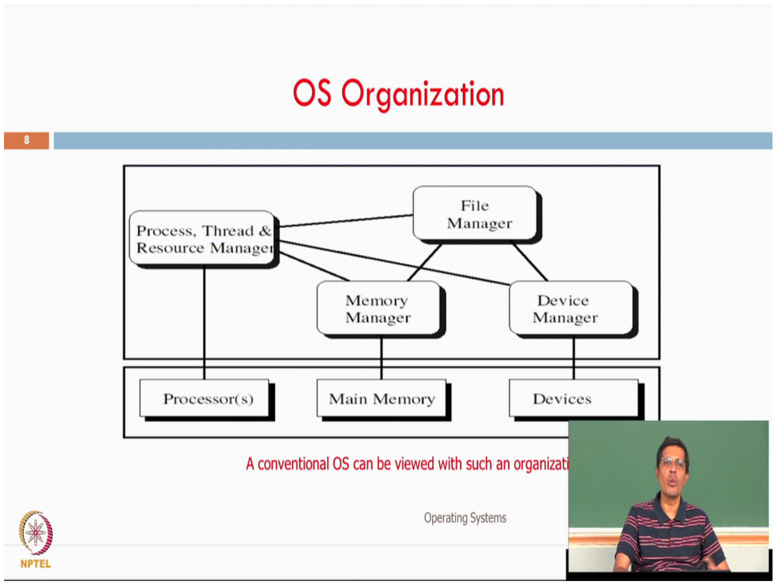
Components/Services of an OS

9

- Due to the complex nature of the modern operating systems, it is partitioned into smaller components.
- Each component performs a well-defined function with well-defined inputs and outputs.
- Many modern operating systems have the following components or provide the following services.
  - Process Management
  - Main Memory Management
  - File Management
  - I/O System Management
  - Secondary Management
  - Networking
  - Protection System
  - Command-Interpreter System

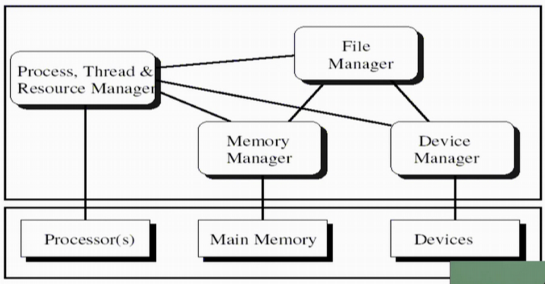
Operating Systems

NPTEL



OS Organization

8



A conventional OS can be viewed with such an organizati

Operating Systems

So to broadly tell, so now that we have an understanding of the operating system to some reasonable extent. These are all some of the services that are expected out of an operating system. The operating system is a very big software. Now to understand operating system, we will take a modular approach wherein, we will break this software into small small components and they start talking to each other.

Already we have seen partition of the different modules in the previous slide as I am just projecting it here. So we have at least four managers managing four important aspects of the operating system. Now many modern operating system have this 8 management modules namely, process management, main memory management, file management, IO system management then secondary storage management, networking, protection system, command-interpreter system. So these are all different components of an operating system and we will see that in better detail in the next module. Thank you.