Introduction to Operating Systems

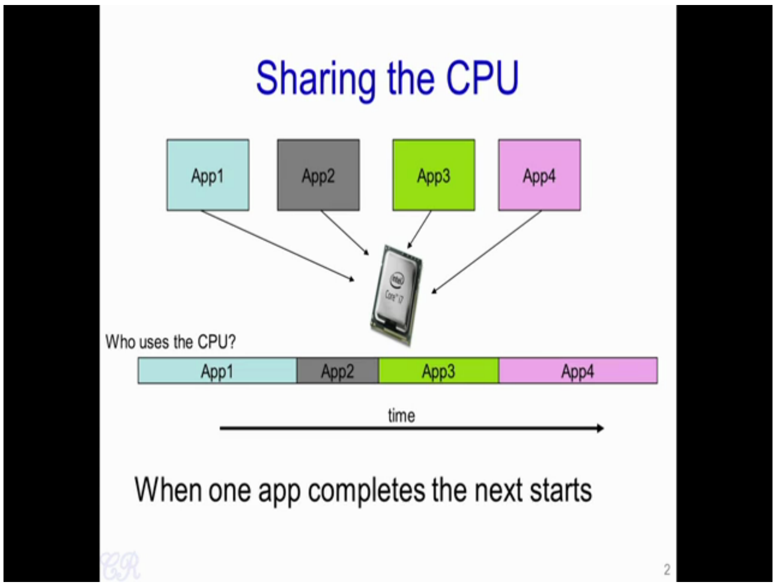
Prof. Chester Rebeiro
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week - 01
Lecture - 04
Sharing the CPU

Hello. In the introductory video, we had seen that one of the most important applications of an operating system is for resource management. Essentially, since the system that the OS runs on as limited amount of hardware resources, so the operating system is in charge of utilizing these resources in an efficient way.

Among all the resources that are present in the system probably the most critical resource is the CPU. So, most systems have a very few number of CPUs; earlier systems in the 1990s and the early 2000s had a single processor, while systems these days are equipped with 4, 8 or 16 CPUs. So, it is important that the operating system ensures that these CPUs present in the system are adequately utilized. So, we will see how these CPUs are managed by the OS.

(Refer Slide Time: 01:32)



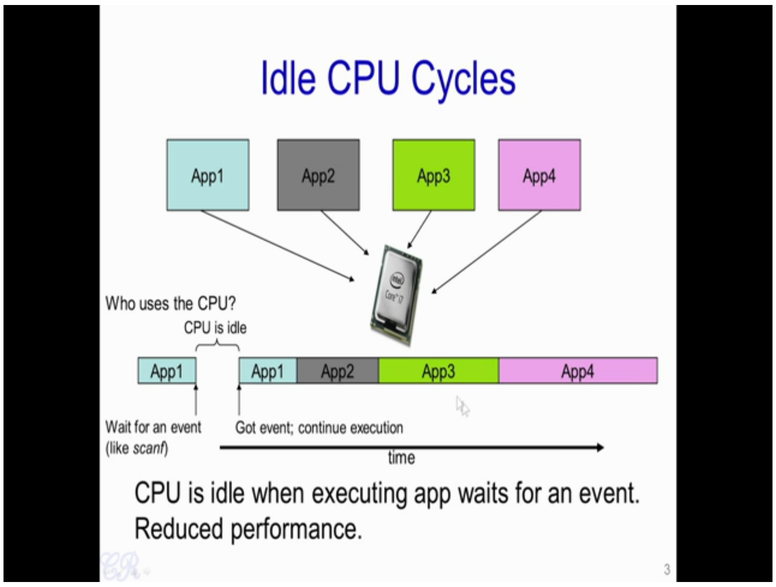
Sharing the CPU

When one app completes the next starts

In systems these days, we have several applications that are running contiguously. So, we had seen that we could have something like office application, internet explorer, a Skype application and a power point application all of them need to share this single CPU. So, how is this possible essentially how does the operating system decide who should run or rather which application should run on the CPU.

One very simple thing that the operating system can do is to put one application onto the CPU, and let it run till it completes. For instance, very primitive operating systems such as the ms dos operating system would start an application, and make it executive in the CPU until it completes. When that application completes then the next application that is app 2 would executive in the CPU till it completes, then application three would start and executive in the CPU till it ends and then application four would start. Well, this is a very easy way to manage the CPU time, it is not very efficient, and we will see why.

(Refer Slide Time: 03:10)



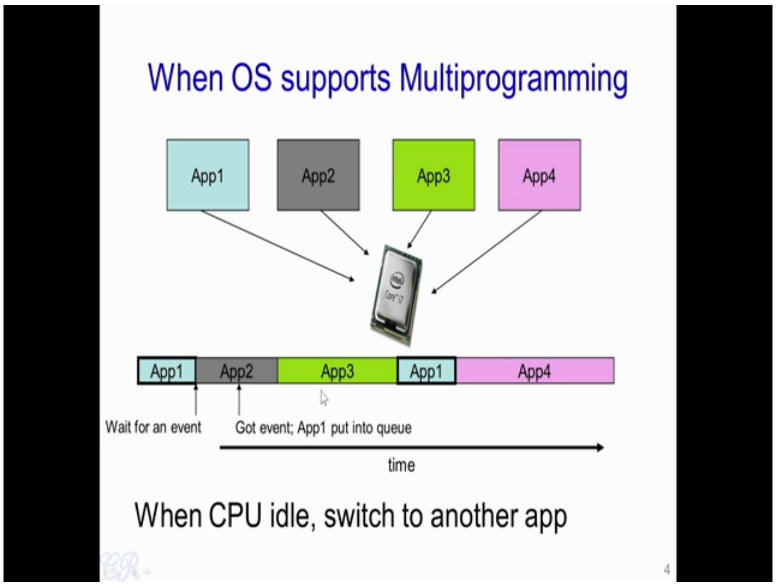
Idle CPU Cycles

CPU is idle when executing app waits for an event.
Reduced performance.

Let us say application 1 is executing on the CPU and after sometime it waits for a particular operation to be done by the user, for instance, the application one is waiting for an event like a scanf. Essentially it is waiting for the user to input something to the keyboard. So, form this time onwards until the user actually presses a key the CPU does nothing, but is waiting idly. Essentially it is wasting time, so these are the idle cycle times of the CPU; only when the event is obtained will the application 1 continue to execute.

Thus, we see that even though this scheme of executing 1 application after the other completes is very simple, yet it does not utilize CPU time adequately or efficiently, because every time that the application requires an event, the CPU would be idle until that event actually occurs. And this as we see would cause a reduced performance with respect to the utilization of the CPU.

(Refer Slide Time: 04:46)



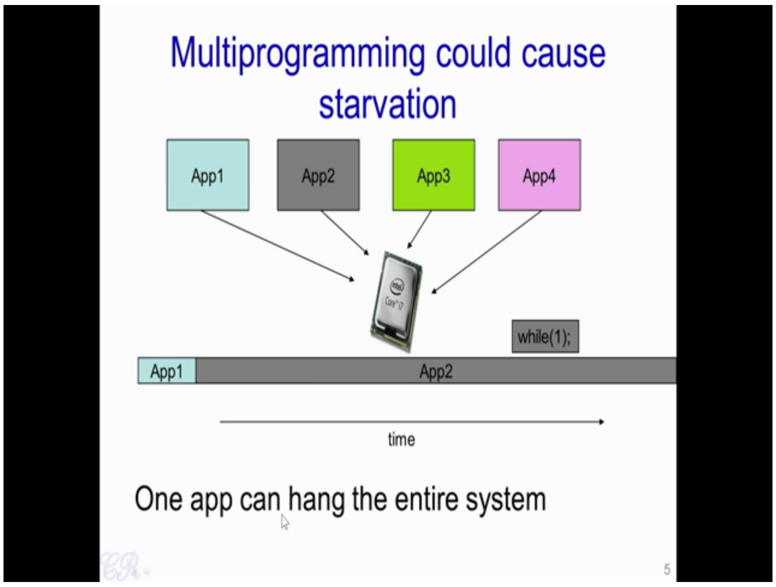
When OS supports Multiprogramming

When CPU idle, switch to another app

A better way to do is something known as multiprogramming. So, in an operating system that supports multiprogramming, the application 1 will continue to executive until it requires an external event. So, when a function like scanf gets executed in app 1 that application will become blocked, essentially it will be preempted from the processor, and another application in this case app 2 will executive in the CPU. After a while when the user inputs something through the keyboard the event is obtained, and as a result of this application 1 is put into a queue.

At a later point in time, application 1 will be again given the CPU and will executive from where it has stopped. Essentially it had stopped during the processing of the scanf, and it will continue to execute from where it had stopped, essentially now has got the input character, which the user has pressed on the keyboard, and it will continue to executive form this point onwards. Therefore, we have seen that we are preventing the CPU form being idle essentially by executing another application on the CPU, we have preventing the CPU form idly running, and therefore, increasing performance. However, there is a problem with this also.

(Refer Slide Time: 06:41)



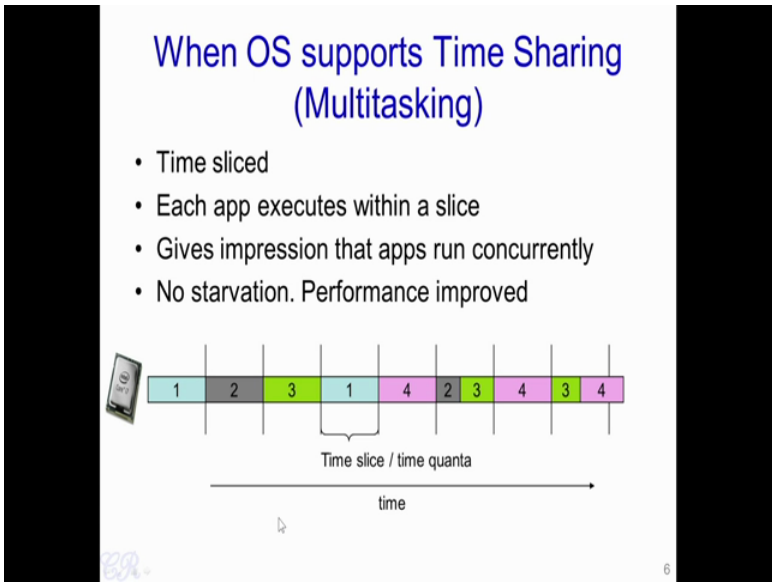
Multiprogramming could cause starvation

One app can hang the entire system

Now, consider this particular scenario where application 1 runs for sometimes and then gets blocked. However, application 2 has something like this are present in its code essentially this is an infinite loop. So, while one would continue to executive infinitely. So, as a result of this app 2 will never stop, it will keep executing and holding onto the CPU.

Due to this another application such as app 3, app 4 and as well as app 1 when once it has obtained the event will never be able to run, the only way to terminate this particular thing would be to forcefully stop application 2 or reboot the system. In earlier operating systems like ms dos, this use to happen quite frequently. As a result of this, it was often require that the CPU be restarted because of one application that is hanging the entire system.

(Refer Slide Time: 07:55)



When OS supports Time Sharing (Multitasking)

- Time sliced
- Each app executes within a slice
- Gives impression that apps run concurrently
- No starvation. Performance improved

More recent operating systems support what is known as multitasking or time sharing. Essentially, in this case, the CPU time is split into slices or so each slice is known as a time slice or a time quanta. In each time slice, a process would run; the process will continue to executive until one of two things happen.

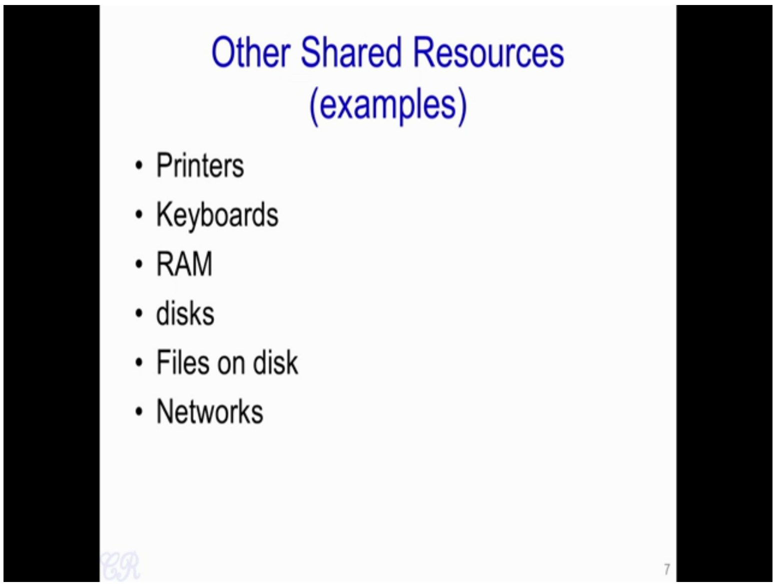
First, it will execute until its time slice completes; in which case another process is given the CPU. So, for instance, over here process 1 executes until its time slice completes, and then process 2 is given the CPU and the process 2 would execute. Then when process two completes its time slice, process 3 will be given the CPU and will continue to execute. Now, in this way all processes are sharing the CPU. After sometime the processes could continue to execute from where it had stopped. For instance process one had executed till its time slice and after sometime based on some decisions by the operating system process one would be given the CPU again to execute. At this point in time process one will continue to execute from where it had stopped.

From a user's prospective, this delay as a result of multitasking that is as a result of other processes being executed or rather said another way, the delay or the intermediate execution of a single process is hardly noticeable because the time slice is very smaller.

The other way that a process stops executing is when it requires to wait for an event as we have seen in the previous slide or it terminates.

In either way, it results in another process being offer the CPU and the new process will execute. So, the advantage of multitasking is that from a user's prospective, it gives the impression that all the applications are running concurrently, there is no starvation. Also over all from a system prospective the overall performance is improved.

(Refer Slide Time: 10:52)



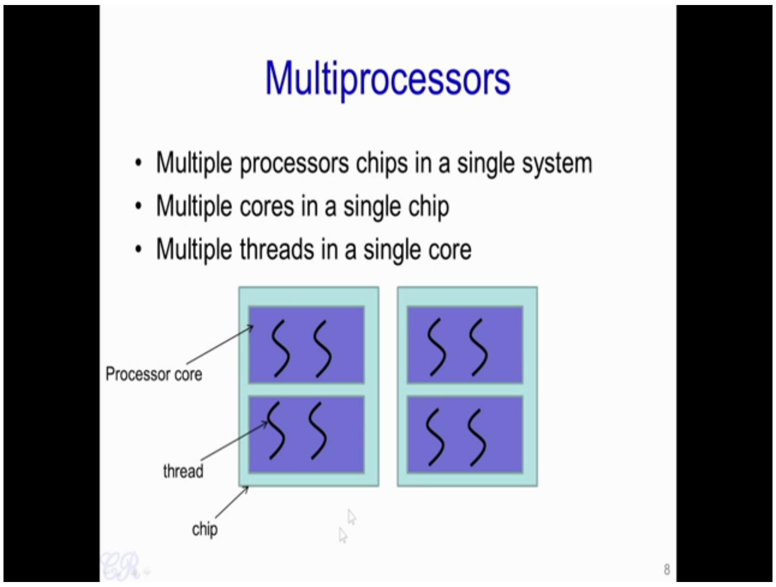
Other Shared Resources (examples)

- Printers
- Keyboards
- RAM
- disks
- Files on disk
- Networks

CR 7

The CPU was one example of the resource which the operating system managed. So, essentially the operating system decided which of the several processes would be given a chance to execute in the CPU. So, along with the CPU, the system would have several other resources; and the operating system would need to share these resources among the various users that are among the various applications. For instance, the printers which are connected to the system, the keyboards, RAM, disk the files on the disk as well as networks. All these are resources in the system and the operating system should manage these resources and ensure that all applications have a fair usage of these various resources.

(Refer Slide Time: 11:54)

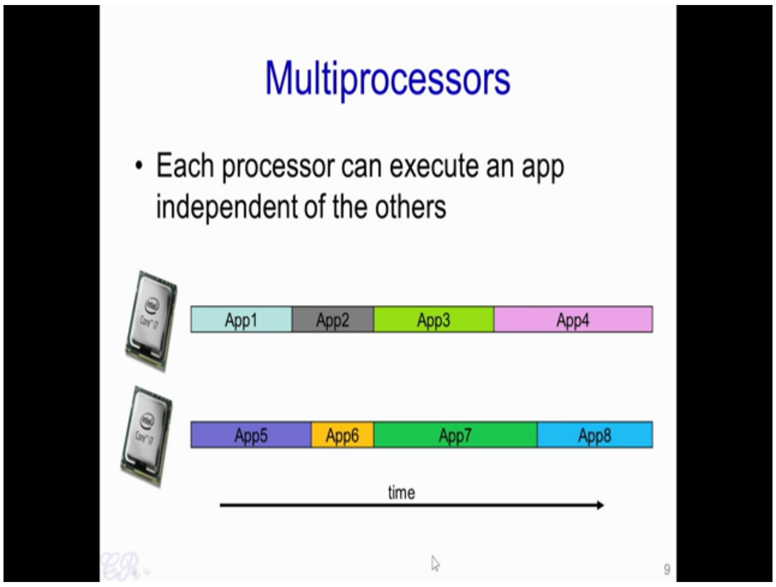


Multiprocessors

- Multiple processors chips in a single system
- Multiple cores in a single chip
- Multiple threads in a single core

So, what would happen in the case of multiprocessors? Essentially, where the system has more than a single processor, such a system would look something like this. Essentially, there would be several chips or CPU chips; and each chip would have several cores. For instance, in this there are two chips CPU chips; and each CPU has two cores; and further it is possible that each core runs multiple threads. So, this is achieved by a technique known as symmetric multithreading; in the Intel nomenclature this is known as the hyper threading. In this particular system, which has two chips total of two cores per chip and a total of two symmetrical threads per core, the CPU has to manage all these resources.

(Refer Slide Time: 13:07)



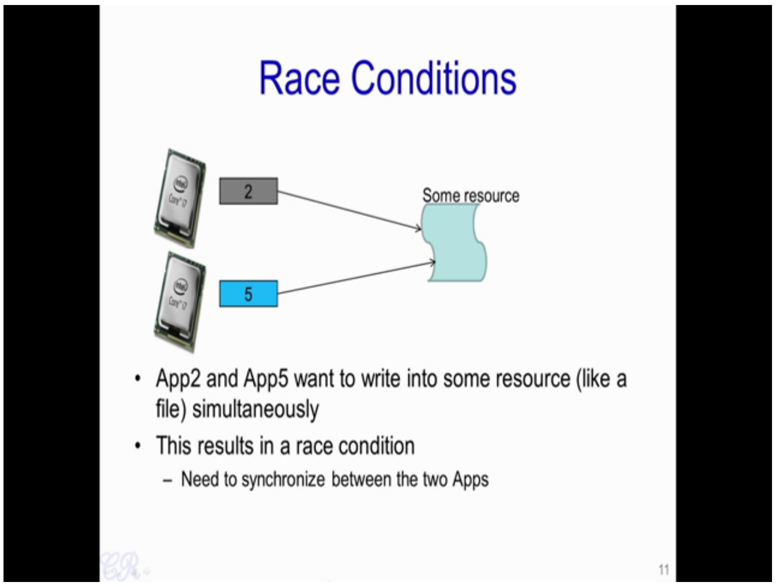
Multiprocessors

- Each processor can execute an app independent of the others

Essentially in such a case, the CPU has to ensure that all computing environments in the system are adequately used. In such a case, we would obtain something has parallelism in the sense that suppose we have two processors over here and let us assume that each processor have a single core in them. So, it would mean that the operating system could then schedule two processors or rather two applications to execute simultaneously on the processors. So, one application will execute on one processor, while the other application executes on the second processor. In addition to this time slicing is possible on each processor that is each processors time could be split into time quanta or time slices as we seen before and shared among the various applications.

In this case, for example, each processor has a time slice which completes periodically; and at start of each new time slice, the operating system running on the processor would ensure that application gets scheduled to execute on that CPU. Similarly, for the second processor, there is also time slices and at the end of a time slice the application being executing will be preempted from the processor and the OS will select another application to execute. So, the operating system should be designed in such a way to ensure that applications unless program to do so, do not execute simultaneously at the same instance in both the processors.

(Refer Slide Time: 15:13)



Race Conditions

2
5
Some resource

- App2 and App5 want to write into some resource (like a file) simultaneously
- This results in a race condition
  - Need to synchronize between the two Apps

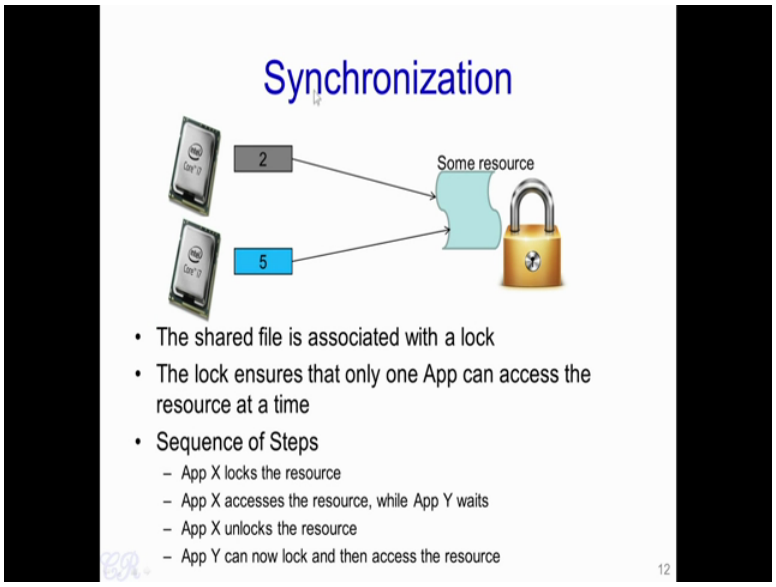
11

Now, one huge issue with a multitasking environment, where time of each processor is split into quanta or time slices allowing different processors or different applications to run concurrently. And also when there are multiple processors present in the system, allowing two or more processors or applications to execute in parallel. So, the issue what arises over here is when two processors or two applications simultaneously request to access some resource.

For instance, let us say this resource is a particular file or it could even a device like a printer, and we have two applications - application 2 and application 5 want to write or print something into this resource. So, both application 2 and application 5 want to use this particular resource So, this results in what is known as a race condition; and it results in quite a huge issue when we are studying operating systems or when we are actually looking deep into operating systems.

Essentially, in order to avoid such an issue, the operating systems need to synchronize between the two applications. It should ensure that were when one process request for a particular resource like a file stored in the hard disk or the printer, and other application app 5 will not be given permission to write or access that particular resource. Only when application 2 complete using its resource, will application 5 be allowed to use that resource; said another way the operating system will ensure that there is a serialized access to this resource, it will ensure that, not more than one application is using this resource at a particular instant of time.

(Refer Slide Time: 17:46)



Synchronization

2
5
Some resource

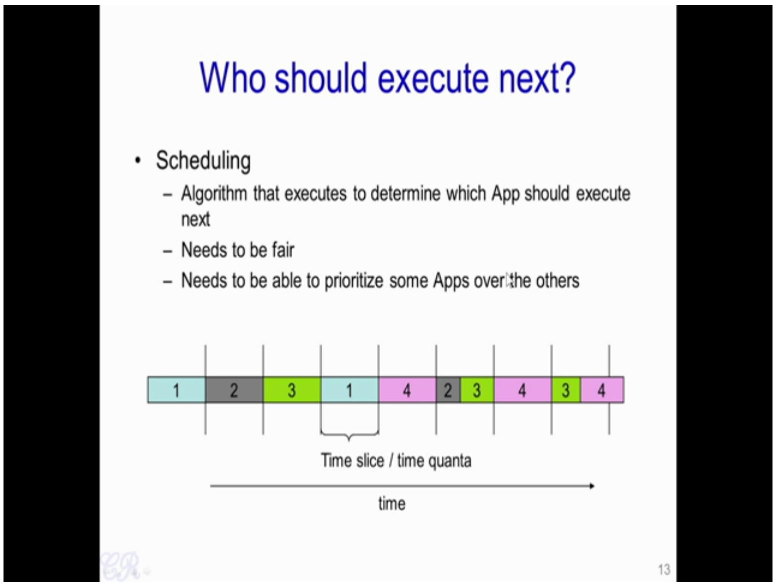
- The shared file is associated with a lock
- The lock ensures that only one App can access the resource at a time
- Sequence of Steps
  - App X locks the resource
  - App X accesses the resource, while App Y waits
  - App X unlocks the resource
  - App Y can now lock and then access the resource

12

Essentially, operating systems solve these problems by using a technique known as synchronization. Now, synchronization you could think of is kind of a lock, which is associated with a resource. So, when application wants to access that resource then it should first get the lock; essentially it should first acquire the lock. So, for instance, we have application X which wants to use the resource; in such a case application X should lock the resource. So, locking the resource would mean that no other application such as App Y or five in this case would be able to use this resource.

Now after application 2 completes using the resource, it will unlock the resource. And during that time, if we have a second application which has also requested for the resource, the second application would have to wait. So, when application 2 completes, it will unlock the resource; and this is a signal to application 5 that it can then use the resource. In order that application 5 uses the resource, it will first lock the resource ensuring that no other application can then use a resource; and at the end of its usage, it will unlock the resource, unlocking will allow other applications to use this shared resource.

(Refer Slide Time: 19:27)



Who should execute next?

- Scheduling
  - Algorithm that executes to determine which App should execute next
  - Needs to be fair
  - Needs to be able to prioritize some Apps over the others

1
2
3
1
4
2
3
4
3
4
Time slice / time quanta
time
13

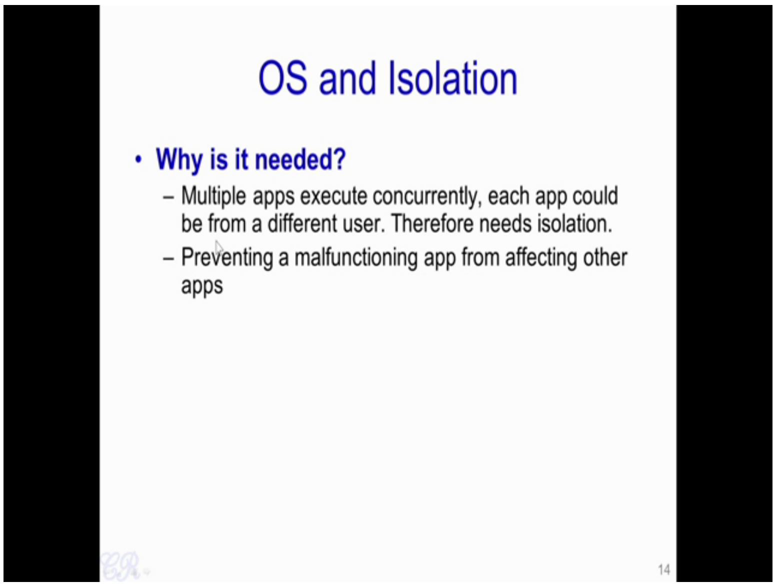
Now, one decision that the operating system needs to make especially in multi-tasking or multi-programmed based system is the decision about which application should run next. So, as we have seen in a multitasking environment application 1 would execute till its time slice, and the operating system then decides which process or which application should run next. So, this decision is made by an entity within the operating system call the scheduler.

Essentially, the scheduler would be designed in such a way that it needs to be fair which means that it should be design in such way that every process or every application running on that system should get a fair share of the CPU. Also in some systems especially, the scheduler should be designed in such way so as to prioritize some applications over the others. And what we mean by this is that we could have some applications which are far more important than other applications, and these applications need to be prioritized. In other words, these high priority applications should be given more CPU time to execute as well as it should ensure that these high priority applications do not wait too long to execute in the CPU.

To take an example, let us say we have three applications - application 1, 3 and 4, which are low priority applications, for instance it could be like executing a compiler or doing some scientific operations like computing the primality of a number and so on. So, these it could consider as a lower priority application. While process 2 let us assume is a high priority application in the sense that it could control some parameter in say robot or it would acquire some information about the environment such as the temperature or humidity of the state of a particular value.

Now, since this application is of a higher priority, therefore the operating system should ensure that this application 2 should be given more CPU time to execute as well as this application should not be waiting too long before it gets the CPU time. So, this entire decisions based on the number of processors as well as the number of CPU cores and the number of threads in each core, and how processor should be executed in which CPU and which thread in the CPU is decided by a scheduler, which executes in the operating system.

(Refer Slide Time: 23:00)



OS and Isolation

- **Why is it needed?**
  - Multiple apps execute concurrently, each app could be from a different user. Therefore needs isolation.
  - Preventing a malfunctioning app from affecting other apps

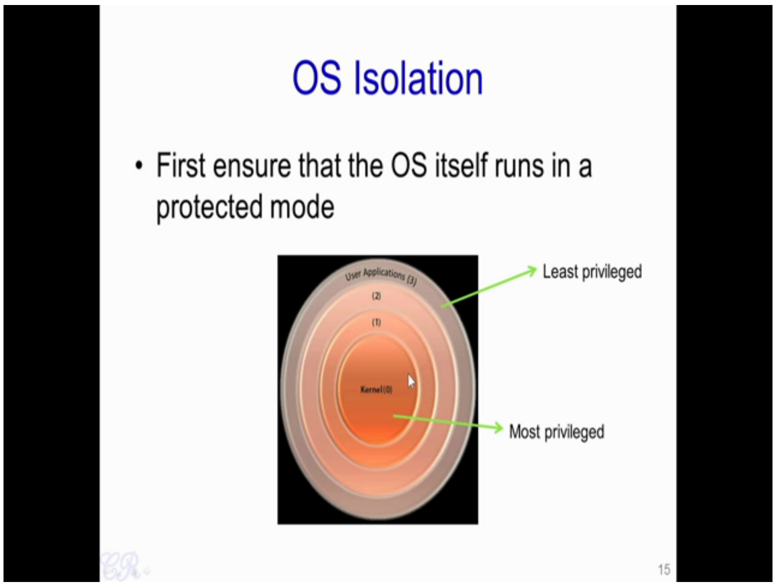
14

And other very important requirement when we speak about operating system is the requirement for isolation. Essentially, this arises from the fact that we would have multiple applications that execute either concurrently or in a time sliced manner in the system. Now these applications could be from different users. Also it could happen that some of these applications are malicious that is these applications may be a virus or Trojan which has manage to find its way into the system and is manage to execute in the CPU. Therefore isolation is required to protect one application from another; and thereby the data in one application cannot be visible from another application.

And other requirement for isolation is with respect to the kernel that is with the operating system. So, the systems are designed in such a way that processors or applications that execute in the system do not directly access various resources.

For instance an application running in the system will not be given direct access to a device such as the printer. If the application wants use the printer, it needs to go through the operating system that is it would need to make a system call which in turn would trigger the operating system to execute device drivers with respect to the printers. The application would then send the document or the text file, which needs to print to the operating system, which then transfers it to the printer.

(Refer Slide Time: 25:08)



OS Isolation

- First ensure that the OS itself runs in a protected mode

In order to achieve this, most processor and most operating systems have this ring like structure. Essentially, in the Intel platform that is the Intel processors, there are 4 rings - ring 0, ring 1, ring 2, and ring 3. Now, ring 0 is where the operating system executes

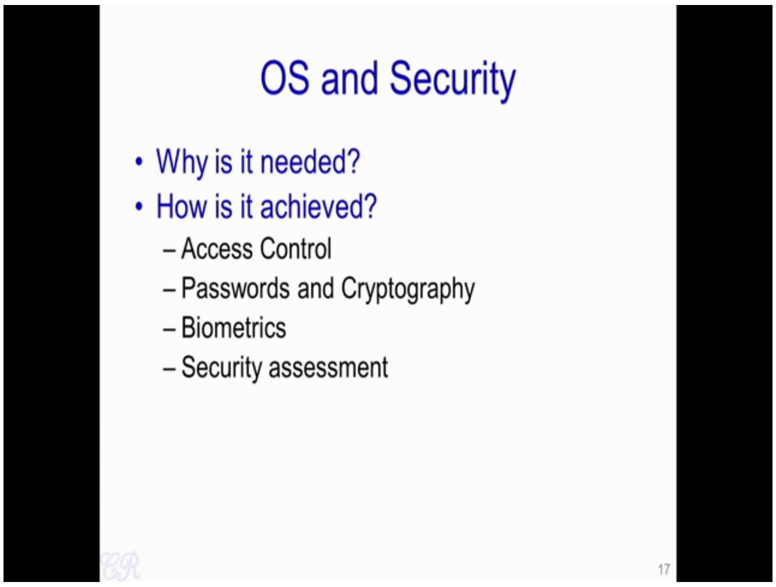
So, this is the Most Privileged mode of operating. In ring 0, the operating system or the kernel which executes over here can do quite a few things like manage the various resources, directly communicate with various devices, and also could have control about the various applications that are executing. In modern operating systems like Linux, which run on Intel processors, the third ring that is ring 3 is used by the user application. So, this is the Least Privileged ring and it ensures that applications that are executed in this ring do not have access to the kernel that is an application running in this ring under normal circumstances will not be able to review or determine anything about the operating system.

So, whenever you run a application on your system such as let us say a web browser or a office application, the application would be started as a user application in ring 3, and in order to use a resource such as the network or to display something on the monitor, the these applications would need to invoke the kernel through the system call. Therefore, the kernel isolated from all the user applications.

In addition to this, the operating systems as well as the processors are designed in such a way that each application that runs in the user space is isolated from each other. What this means is that if we have say of a web browser running here, and also an office application, the entire system of the operating system along with the processor will ensure that the web browser has no information about the other application that it is it does not know what the office application is doing. And the office application will not know what the web browser is currently executing. For instance, the office application will not know what web pages are currently being browsed.

So, this you could see is useful to isolate different users as well. So, if you have two different users who are sharing the system, so user 1 will not know what the user 2 is executing unless he makes use of certain kernel functions. And the only way to use those kernel functions is through system calls or in some certain cases like Linux, there are certain functionalities which are displayed about the kernel. So, the user will then able to use those functionalities of the kernel to determine about the other process.

(Refer Slide Time: 29:09)



OS and Security

- Why is it needed?
- How is it achieved?
  - Access Control
  - Passwords and Cryptography
  - Biometrics
  - Security assessment

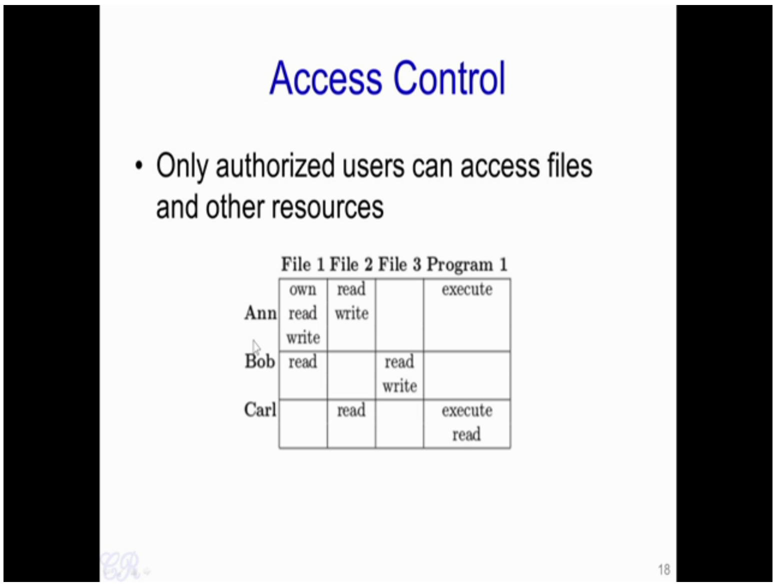
17

Another important aspect when we are talking about operating systems is the need for security. Essentially, security features are added to operating systems to ensure that only authorized users are able to use that system. So, if for instance, I do not have user account in a particular server, then the operating system will ensure that I will not be able to login to that particular server, and I will not be able to run any application in that server. So, how is security achieved in the system is by several features that operating systems support. For instance by having a techniques know as access control, which will ensure that some files that are created by a particular user is not visible to another user.

For instance, suppose I login to a system as a guest that is through a guest account. So, the operating system will then know about this and determine what files I am allowed to execute and what files I will be able to access. So, one way to achieve security is by using something knows as access control. So, access control mechanisms in the operating system would allow or manage the various resources. For instance, if I login to a system using a guest account then the operating system will detect this, and determine what are the resource and what are the files that I can access.

For instance, in normal guest account, I may not be able to access let us say the USB drive, while on the other hand, I may be have given read only access to certain files present in the hard disk. While other files which are system related or based the operating will not even be given read, write So, I will not be able to view those files at all. Another technique, which is used by the operating system in order to achieve security, is the user passwords and cryptography. So, passwords ensure that unauthorized people are not allowed to use this system. So, modern laptops also use techniques like biometrics or the figure prints scans to filter people who are given access into this system.

(Refer Slide Time: 32:00)



Access Control

- Only authorized users can access files and other resources

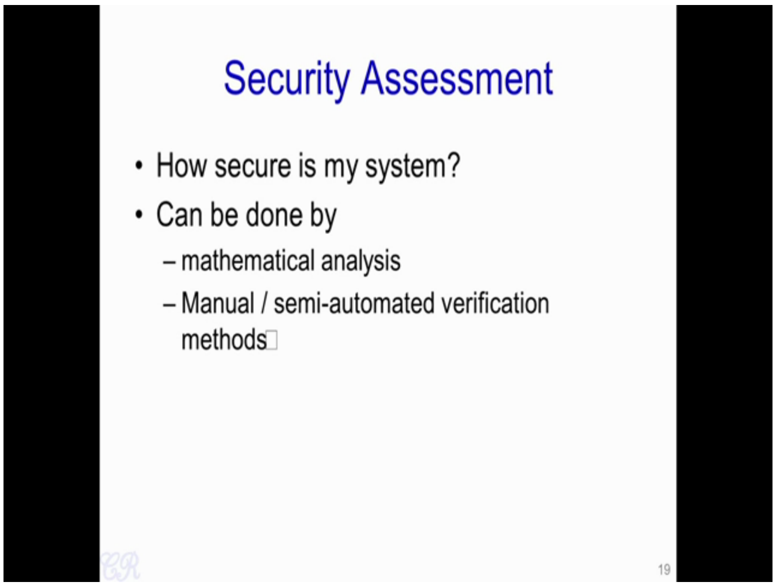
	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read

18

This particular slide shows how access control is implemented in typically. So, essentially we have a matrix here and each row corresponds to user. So, Ann, Bob and Carl are three users of the system, while the columns show the resources that is file 1 which is stored in the hard disk, file 2, file 3 and program 1. So, in the system the operating system depending on the user who has logged in based on their password is given different permissions for each file.

For example; The user Ann has the read and write permission for file 1 as well as file 2, and can execute program 1; however, Ann has no permission at all for file 3. In a similar way, Bob another user of the system can only read file 1, he cannot write to file 1, but he can read as well as write to file 3, and based on permission said in the operating system, bob does not have the access to program 1 that is he cannot execute the program 1. Similarly, the third user Carl cannot have access either read or write access to file 1 and file 3, and can have only access to file 2; however, Carl can execute program 1 as well as can read program 1.

(Refer Slide Time: 33:46)



Security Assessment

- How secure is my system?
- Can be done by
  - mathematical analysis
  - Manual / semi-automated verification methods

Let us say now that we have our system filled with the processor, the various resources and the operating system which manages these various resources and also manages the applications that execute on the system. So, how do we access the security of the system? So, there are two ways in which this is generally done. This is done by mathematical analysis or by manual or semi automated verification techniques.

So, many systems especially those which are used in critical applications such as the military or other defense related applications, go through regress security assessment to ensure that these systems including the operating system as well as the hardware and applications have sufficient amount of security, so that it could be used for such critical applications. So, this is more of a testing and validation related aspect in the system.

Thank you.