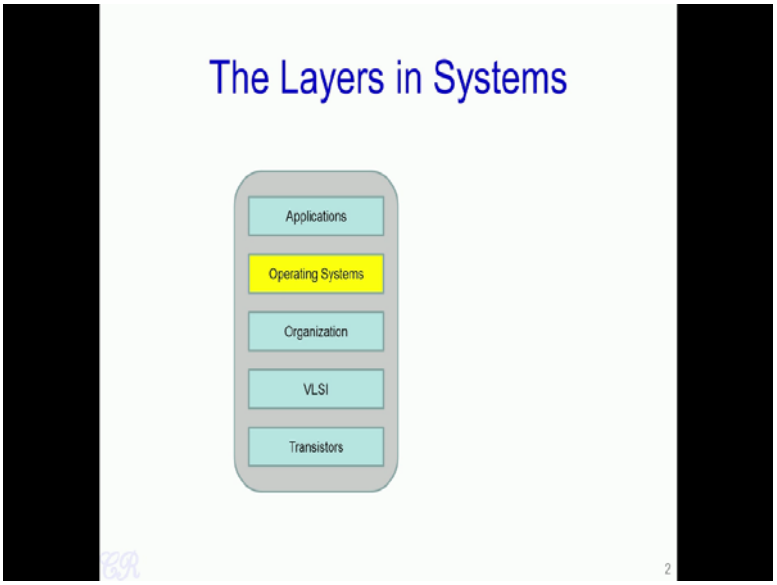
Introduction to Operating Systems
Prof. Chester Rebeiro
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week – 01
Lecture - 01
Operating Systems (An Introduction)

Hello and welcome to the first weeks lecture for the course An Introduction to Operating Systems. This week we will be building a platform for the course upon which the following weeks lectures will depend upon. In this particular lecture, we will give a very brief introduction to OS. In particular, we will look at where the operating system actually fix in the entire computer and also, we will see what the essential role of the operating systems in the computer.

(Refer Slide Time: 00:57)

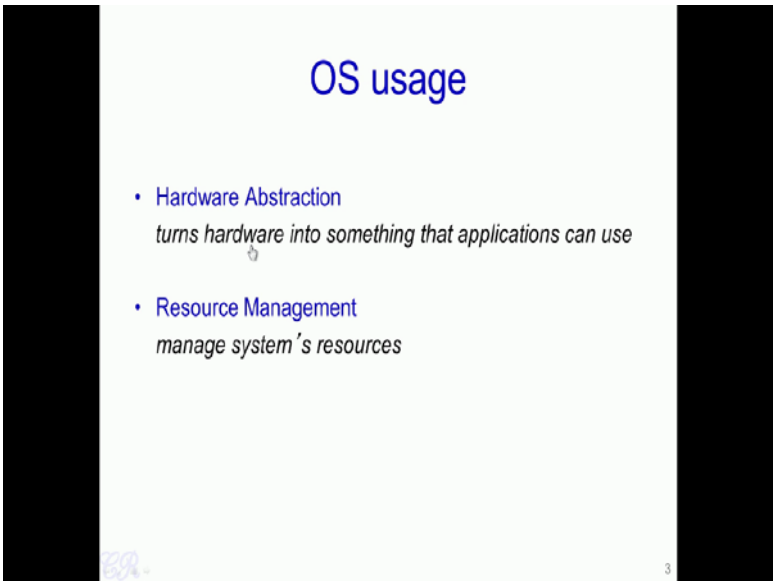


So, when we look at computer systems, we can think of it in different layers. So, at the bottom layer are millions and billions of transistors. These transistors are typically CMOS transistors and these transistors are then composed together to build several logical gates and these gates would include several digital logical gates, such as gate or x or and so on. Included in this particular layer are various things like the memory cells, flip flops registers and so on.

Now, all these basic VLSI units are then organized into various forms to build things like the memory RAM, the decode unit, the instruction fetch unit and so on. So, this organization as the third layer in the computer system is what we actually see as the hardware. So, this is the physical hardware that we actually purchase from the store. Now, when we purchase this computer hardware, we could execute several different applications for instance we could have office applications or internet explorers and so on.

Now, setting between these applications as well as the hardware is the operating systems. So, the operating system essentially would manage both the applications that execute on the computer as well as it would manage how the resources are utilized in the system. So, let us see more in detail how the operating systems are used in the entire scheme.

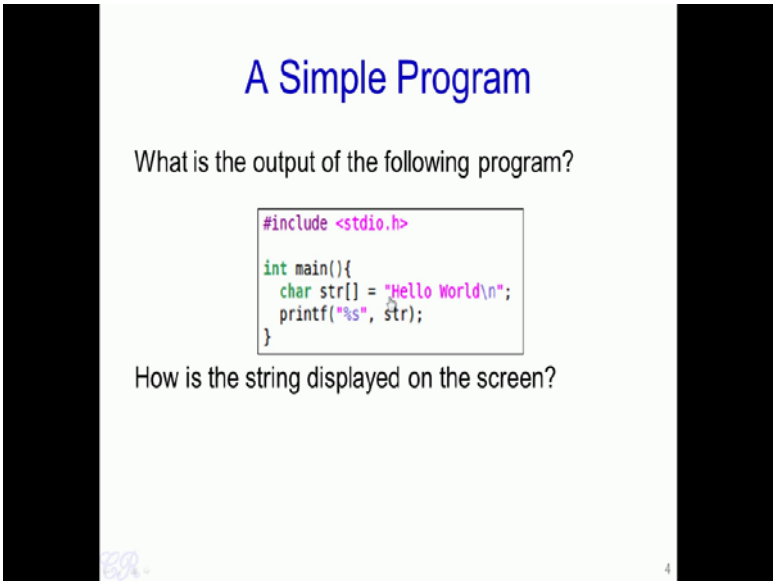
(Refer Slide Time: 02:51)



To look at it broadly, the operating system is used for two things. It first provides hardware abstraction and second manages resources in the system. The hardware abstraction essentially is used in order to turn the hardware into something that software applications can utilize easily, while the resource management is required because of the limited resources that are present in the computer.

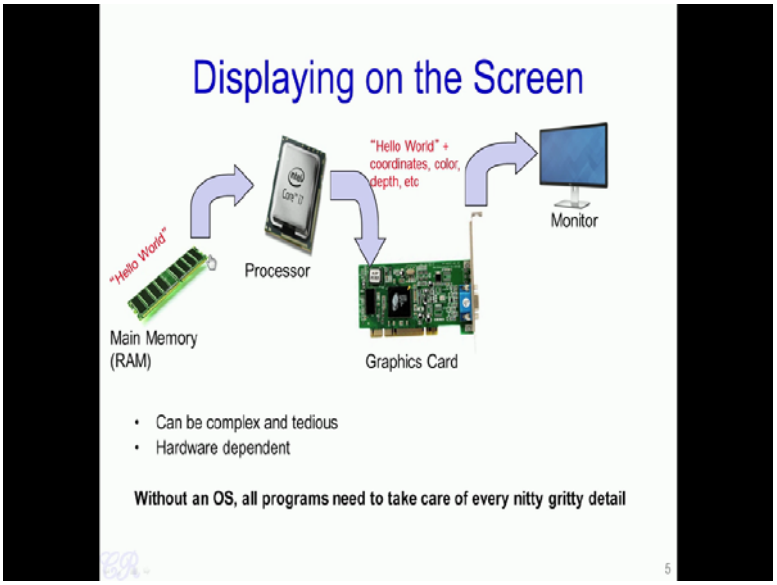
So, we will look at these two uses of the operating system in more detail.

(Refer Slide Time: 03:31)



Let us start with this very simple program. So, this is a program written in the c language and essentially it is going to print this string “Hello World” on to the monitor. So, this is a very simple program and essentially this string “Hello World” is stored in memory and it is pointed to by this pointer str. Now, printf is passed this pointer s t r and would result in this string being printed on to the monitor. Now, the question which comes is how exactly is the string displayed on to the monitor, what is the process involved to get this string which is stored in memory to be displayed on to the monitor.

(Refer Slide Time: 04:25)



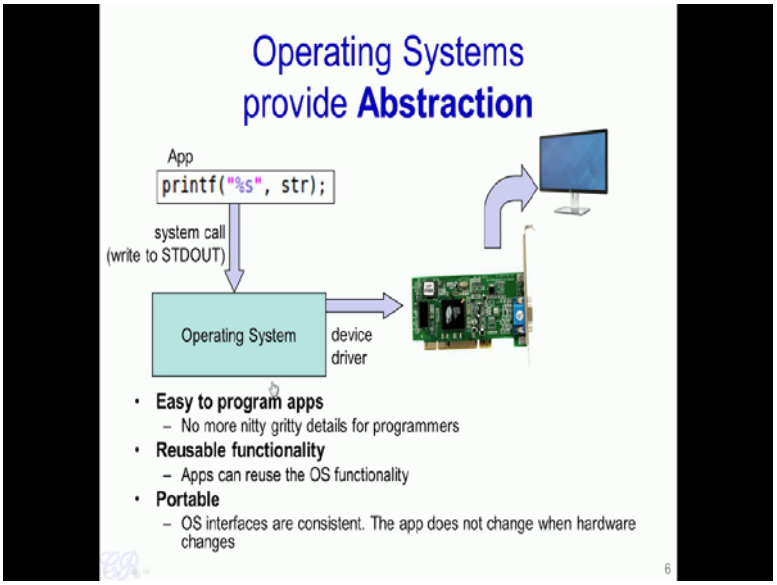
So, in fact when we look at the entire scheme, the string “Hello World” would be present in some memory location in the main memory which is also called as the RAM. Now, there are certain instructions in the processor would read this string byte by byte from the main memory into registers and then, copy them on to something known as a video buffer along with copying the string “Hello World” to the video buffer. Other attributes are added for instance things like the color of the string to be displayed, the x y coordinates on the monitor whether “Hello World” string needs to be displayed and other monitor specific attributes such as the depth and so on.

Now, this string which is copied to the video buffer is then read by the graphics card which would then display it on the monitor. So, this is the entire process of displaying a string “Hello World” on to a monitor. So, now you would see that doing this is not trivial. It is in fact quite complex as well as tedious. Imagine that every program that you write would require to do all these things like knowing where in the memory the “Hello World” is stored and then, how to actually display it on to monitor, how to compute the coordinates, how to specify the color, the depth, another attributes and how exactly to pass this information to the graphics card and so on.

Another aspect is that this is extremely hardware dependent any change in one of these

things. For example, if the processor changes or if the graphics card or monitor changes, then it is quite likely that the program will not work and for example, if the monitor changes, then there may certain attributes which need to be specified differently for the new monitor or if the graphics card changes, then perhaps the way the coordinates are set where a depth and color are set for this string “Hello World” would differ. Therefore, without the operating system, every programmer would need to know about the nitty-gritty details about the hardware. Essentially he would need to know what hardware is used in the system and also, how the various aspects about the hardware fit with each other.

(Refer Slide Time: 07:15)



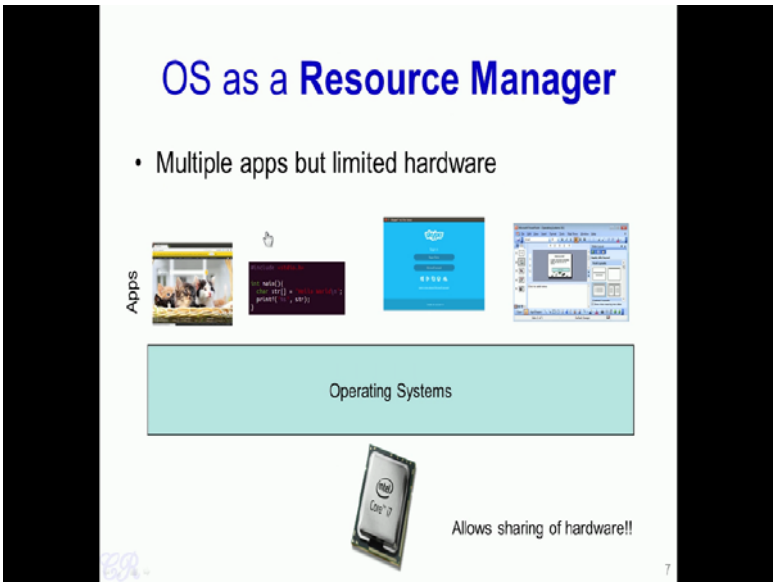
Operating Systems essentially provide Abstraction. They sit in between the applications that we write and the hardware and abstract all the nitty-gritty details of the hardware from the applications. So, simple statement such as this printf, str would eventually result in something known as a system call which would trigger the operating system to execute the OS will then manage how this string str will be displayed on to the monitor, everything such as how the color needs to be set, how the x y coordinates need to be set and so on. It would be done internally by the operating system. So, from an application prospective and from a programmer prospective, all these details are abstracted out.

So, as you would see this would make writing programs extremely easy. So, the programmer need not know the nitty-gritty details about the hardware any more. Second advantage is the reusable functionality. Essentially applications can reuse the operating system functionality. So, what we mean by this is that since the OS abstracts, the hardware details from the applications, all applications that execute in this system could just reuse the operating systems features.

For example, every application that uses printf will be invoking the operating system and OS will then take care of communicating with the hardware. There is a single module in the operating system which handles all printf's from all applications executing on the system. Third advantage is the portability. Essentially what this means is that when we write a program which uses something like a printf statement, we do not really bother about what hardware it runs on. It could run in a desktop for instance or a server or a laptop or if complied appropriately, also in several embedded devices.

So, the underlying operating systems actually would then distinguish between various hardware, that is present and with them, ensure that this printf would be executed appropriately depending on the hardware. So, what we achieve with portability is that essentially applications will not change even though underlying hardware changes.

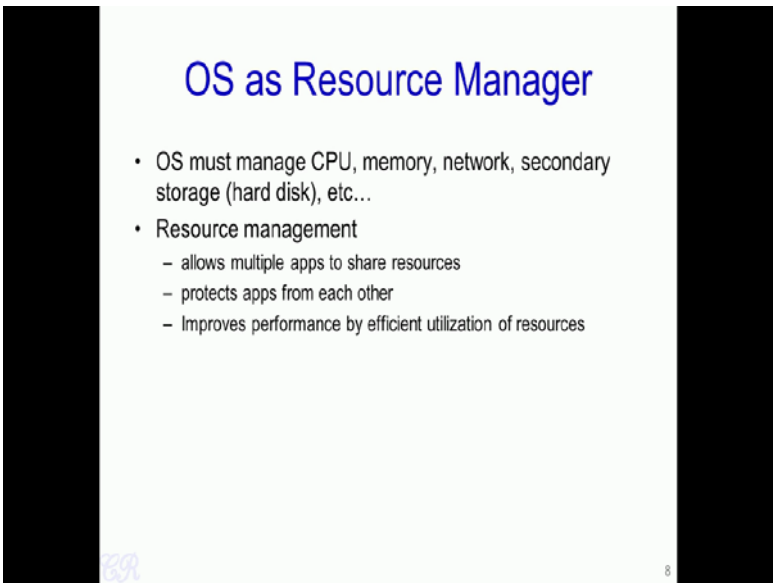
(Refer Slide Time: 10:12)



The second use of the operating system is as a resource manager in the desktops and laptops that we use today. There are several applications which run almost concurrently. For instance, we would be using a web browser to browse the internet and almost at the same time, we would be compiling some of the programs that we have written and we are also executing them or we could be using Skype or a power point application at almost the same time.

Now, the fact is that we could have several applications running on a system, but the underlying hardware is constrained. Essentially, we have just one hardware which needs to create a several applications almost concurrently. So, the operating system ensures that it is feasible for multiple applications to share the same hardware.

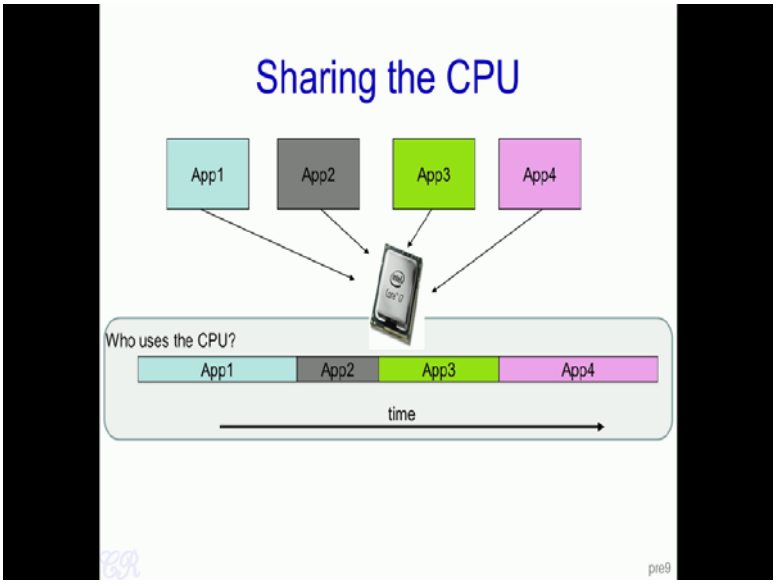
(Refer Slide Time: 11:15)



Now, within this computer hardware, there are several components which the operating system manages. For example, the CPU, the memory network, the secondary storage devices like the hard disk, the monitors and so on, all these components within your computer have a restricted amount. So, you may typically have around 2 or 4 or 8 CPUs present in your system, also the memory may be restricted to 4 or 8 GB, you have typically one network card or one or two hard disk and so on.

So, the operating system needs to manage all these various devices and components present in the system and share these components among several applications almost concurrently. So, with the help of the operating systems, it would be possible that multiple applications share this limited resources and also, the OS is built in such a way that applications are protected from each other essentially the underlying where the operating system is designed as such that every component in the system is adequately utilized.

(Refer Slide Time: 12:35)

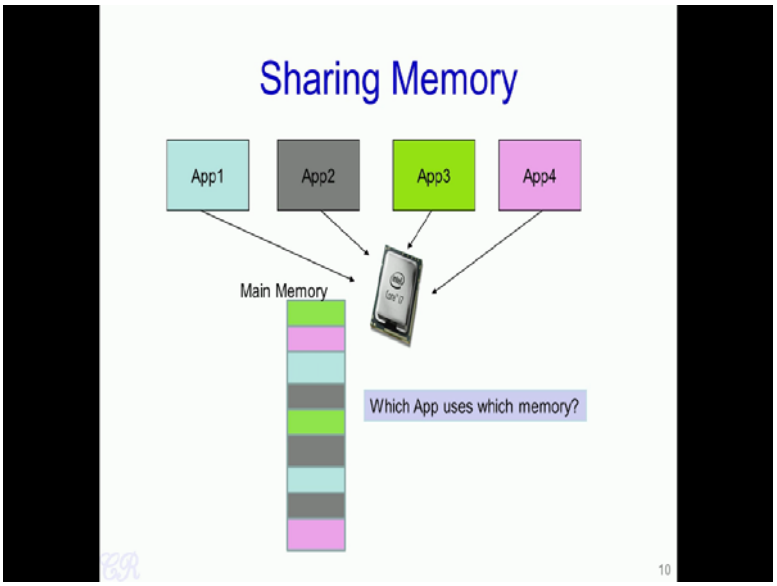


To take an example let us consider the CPU. So, systems typically would have one or two CPUs and multiple applications executing on that CPU. So, how does the operating system share the single CPU among multiple applications? So, one way which was typically done in the earlier operating systems around the late 70s and early 80s was to allow one application to execute on the CPU till it completes and then, start the next application. For example, application 1 is made to execute on the processor and after application 1 completes its execution, only then application 2 is made to start.

Now, this scheduling of the various applications on the processor is managed by the OS. Now, this scheme of sequentially executing one app after the other completes although very simple to implement in the operating system, it is not the most efficient way to do

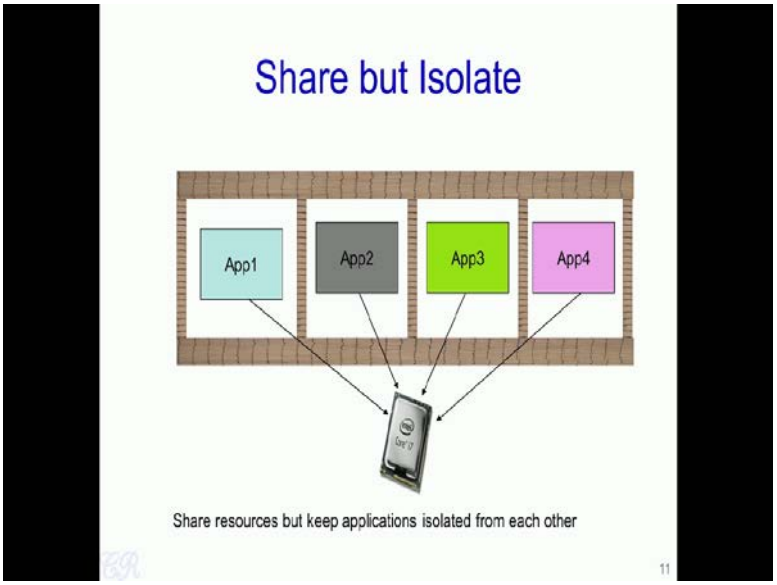
things. So, as we will see in a later video, we will see what the issues with having applications execute in this particular way are and we will see how modern day operating systems manage to utilize this processor in a much more efficient manner.

(Refer Slide Time: 14:13)



Another important component in the computer system that requires to be shared among the various applications almost concurrently is the main memory. Now, in order to execute each of these applications needs to be present in the main memory that is the RAM of the system. The operating system needs to ensure how this limited RAM resource is shared among the various applications executing on the system.

(Refer Slide Time: 14:46)



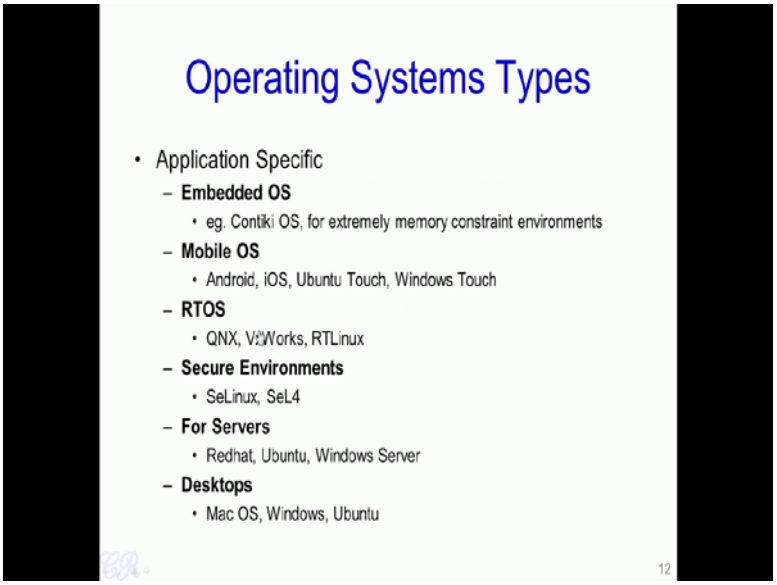
Now, what makes it difficult for the operating system is that in spite sharing the limited hardware resources among the various applications which are executing on the computer system, the sharing should be done in such a way, so that applications are isolated from each other. So, each application, the OS should ensure runs in sand boxed environment that is application1 should not know anything about application 2, application 2 should not know anything about the other applications running on the system and so on.

So, why is this isolation required? To take an example of why isolation is required, let us consider that application 1 is a web browser in which you are doing banking transaction. For example, you are entering your passwords and credit card details in the web browser which is executing as application 1.

On the other hand, application 2 may be gaming software and let us assume that it is having a virus that is it malicious application. Now, assume that we do not have isolation application 2 would be able to determine what application 1 is doing or in other words, the malicious application will be able to determine what is happening in the web browser and therefore, may be able to steel certain sensitive information such as your passwords and your credit card numbers. Therefore, the operating system should ensure that all these applications are isolated from each other and as we can see this is not a very easy

thing to do.

(Refer Slide Time: 16:45)



Operating systems will be quickest. Almost every smart device that we use today has some form of operating system present in it. So, this particular slide shows the various classifications of operating systems depending on the type of application they are intended for need us to say each of these operating systems are designed keeping the application in mind.

For example, the embedded OS such as Contiki operating system or Contiki OS are designed for memory constraint environments, such as wireless sensor nodes that are used in the internet of things operating system like the Contiki OS are designed with the power consumption kept in mind. So, these operating systems manage the various resources and also, abstract hardware in such a way that the power consumed by the entire system is kept minimum. A second class of operating systems which many of you may be familiar with is the mobile operating system or mobile OS.

So, examples of these are the android IO's, Ubuntu touch and Windows touch. So, these operating systems like the embedded OS are designed in order to ensure that the power consumed by the device is Minimum. For example, these operating systems are

designed, so that the battery charge of your mobile phone is extended for the maximum amount of time. So, the mobile OS is like once we mentioned over here has quite similarities in this aspect with the embedded operating systems like Contiki OS.

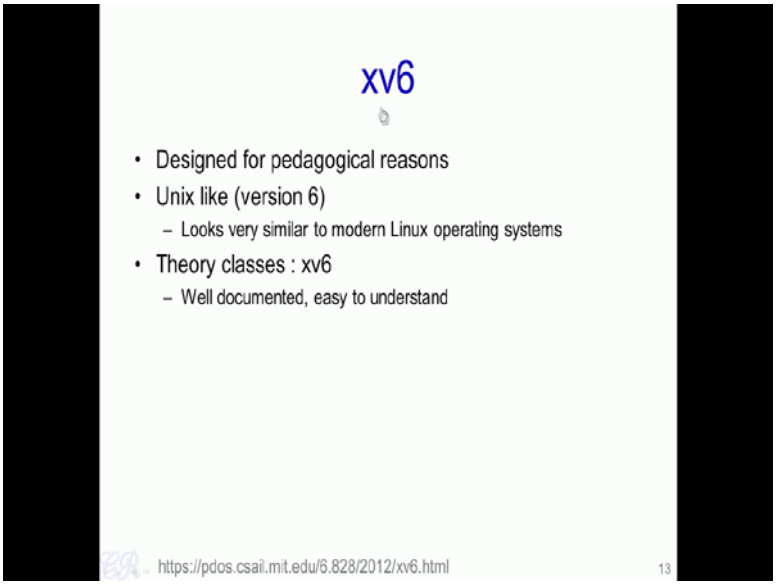
However, mobile operating systems unlike the embedded OS also need to take care of certain special devices, for instance the monitors or the LCD screens and key pads. In other words, the mobile OS also has user interaction which typically is not present in the embedded OS.

Third type of operating system is the RTOS or Real Time Operating Systems. So, examples of these are QNX, VX Works and RT Linux. So, these operating systems are used in machine, critical applications where time is very important. For example, in several machine critical applications like rockets, automobiles or nuclear plans, a delay in doing particular operation by the computer system would be catastrophic. So, these tosses are designed in such a way that every critical operation on the system is guaranteed to complete within this specified time.

Another classification of operating systems, are those used in secure environments. So, examples of these are SE Linux SeL4. So, these operating systems are especially utilized for applications where security is extremely critical.

So, these for example could be for web servers that host banking software and so on. So, the other classes of operating systems which you are quite familiar with are for those used for servers and desktops, such as the Redhat, Ubuntu and Windows server OS's while desktop operating systems are for example Mac Windows and Ubuntu. So, while these two operating systems have several features which are similar, there may be certain differences in the way the OS manages the various applications running on it.

(Refer Slide Time: 20:54)



The operating system that we will be studying for this course is the XV6 OS which is designed by MIT specifically for teaching purposes. So, the XV6 OS is small, well documented and easy to understand. Further, XV6 designed to look similar to UNIX version 6. So, what this means is that the way XV6 is designed is how various UNIX like operating systems like Linux actually works. Therefore, understanding XV6 would give you a nice insight about other modern day operating systems such as Linux.

Thank you.