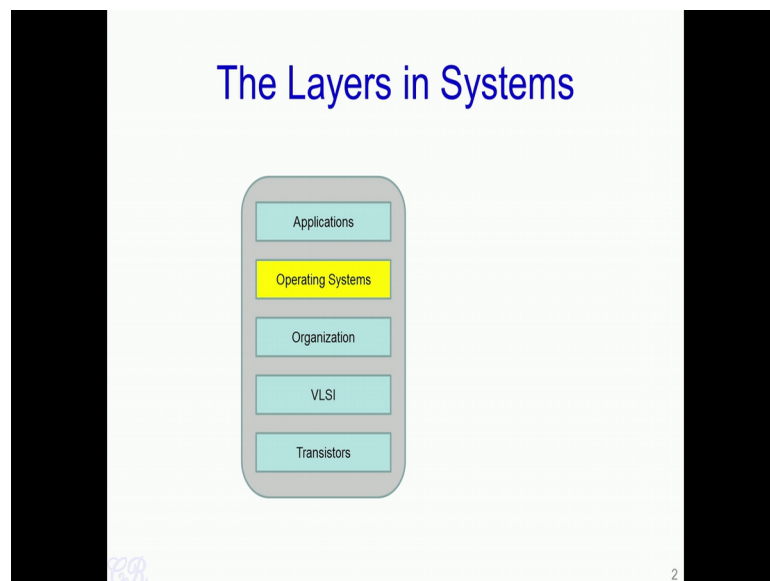


**Introduction to Operating Systems**  
**Prof. Chester Rebeiro**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**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 fits in the entire computer and also we will see what is the essential role of the operating systems in the computer.

(Refer Slide Time: 00:57)

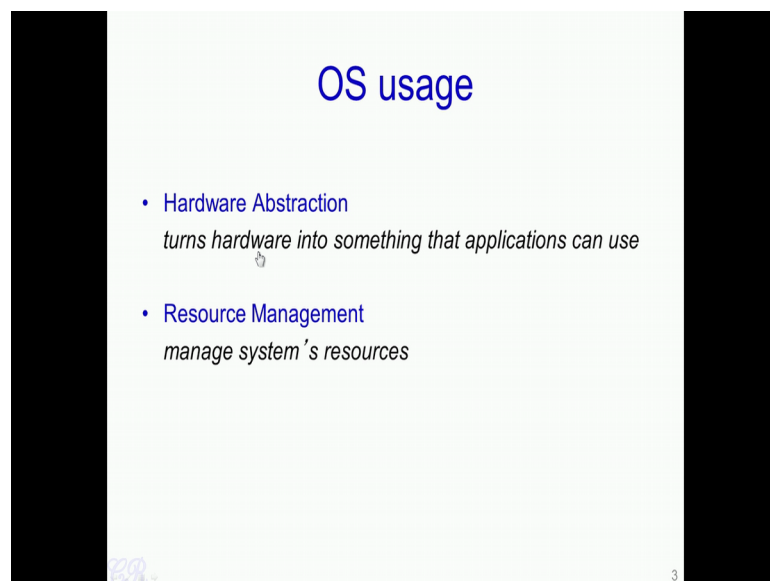


So, when we look at computer systems, we can think of it in different layers. So, right at the bottom layer are millions and billions of Transistors. So, these Transistors are typically CMOS Transistors and these Transistors are then composed together to build several logical Gates and these Gates would include several digital logical Gates, such as the AND gate, OR, XOR 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 (in the slide above) which is 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, sitting 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.

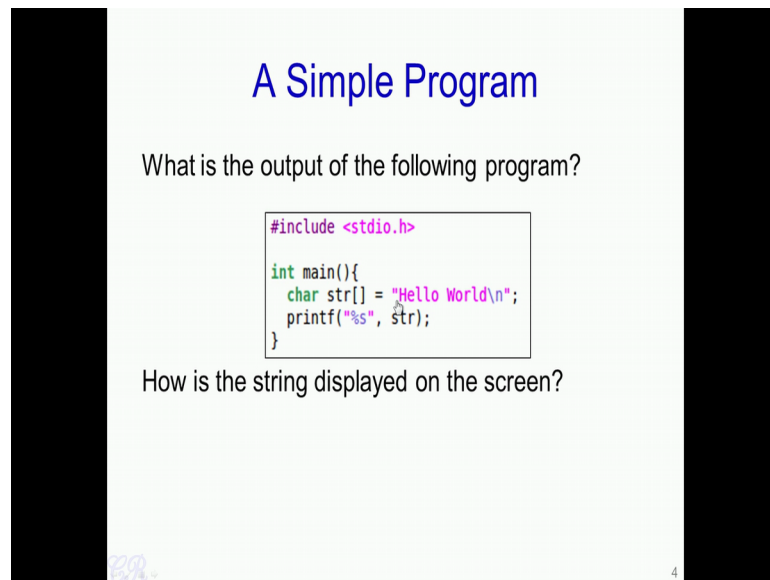
(Refer Slide Time: 02:51)



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)



## A Simple Program

What is the output of the following program?

```
#include <stdio.h>

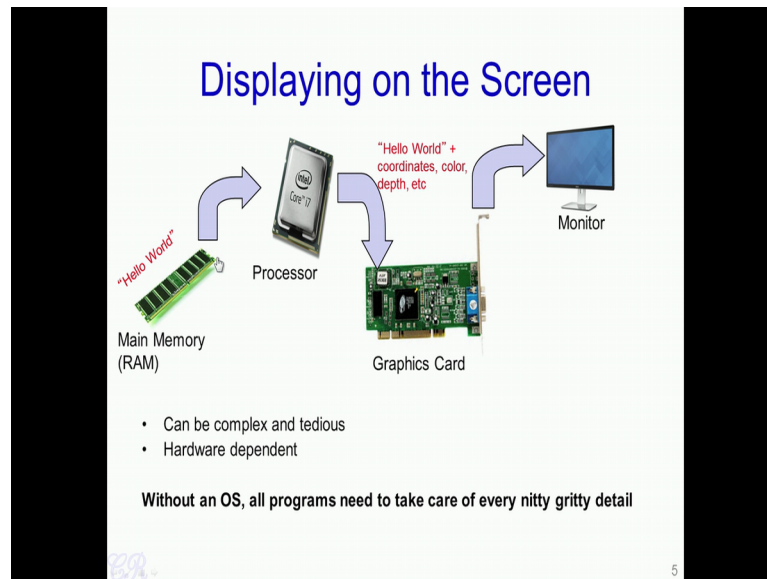
int main(){
    char str[] = "Hello World\n";
    printf("%s", str);
}
```

How is the string displayed on the screen?

4

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 the string “Hello World” on to the monitor. So, this is a very simple program and essentially the string “Hello World” is stored in memory and it is pointed to by this pointer `str[ ]`. Now, `printf` (mentioned in above slide) is passed this pointer `str` and would result in the 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 the string, which is stored in memory to be displayed on to the monitor?

(Refer Slide Time: 04:25)



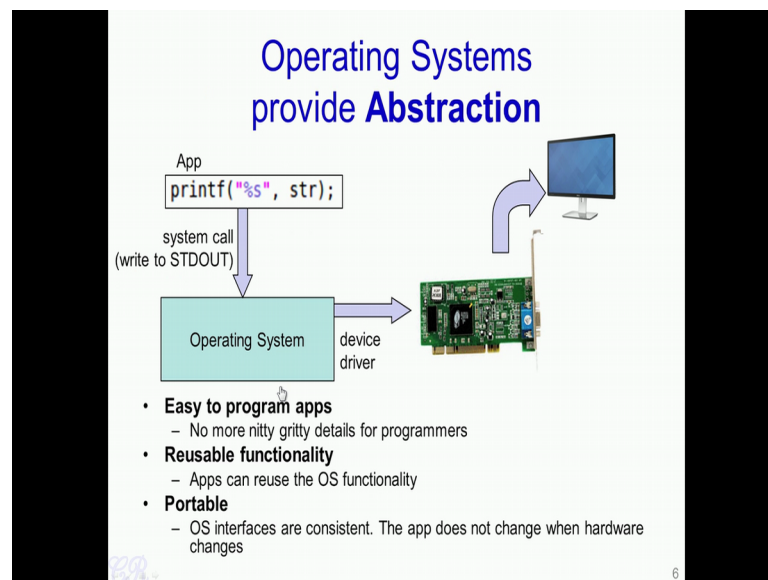
So, in fact when we look at the entire scheme, the string “Hello World” would be present in some memory location in the Main Memory which is also called as the RAM. Now, there are certain Instructions in the Processor would read the String byte by byte from the Main Memory into Registers and then copy 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 where the “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 the Monitor, how to compute the coordinates, how to specify the color, the depth and other 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. For example, if the Monitor changes, then there may be 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 the 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)



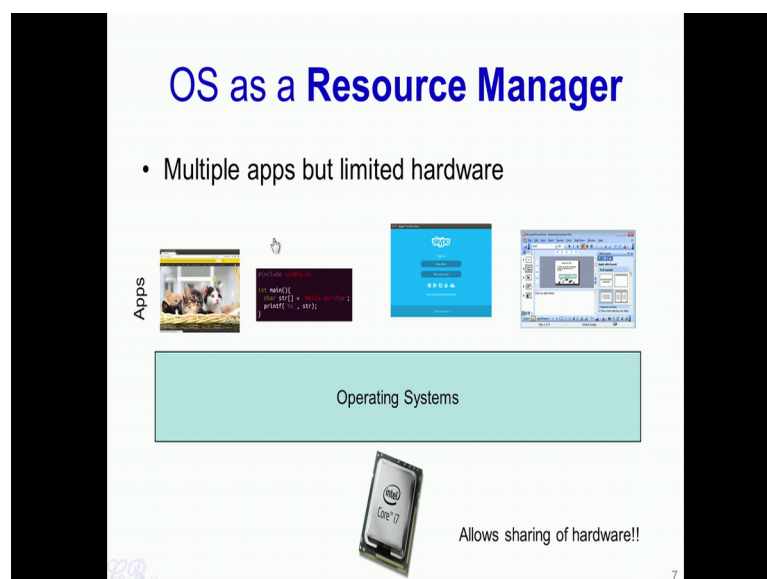
So, Operating Systems essentially provide Abstraction. They sit in between the Applications that we write and the Hardware, and abstract all the nitty-gritty details of the Hardware from the Applications. So, a simple statement such as this `printf("%s", str)` would eventually reside in something known as a system call which would trigger the Operating System to execute. The OS will then manage how the 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 would be done internally by the Operating System. So, from an Application perspective and from a programmer perspective, 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. A 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 the OS will then take care of communicating with the Hardware. There is the single module in the Operating System which handles all printf's from all Applications executing on the System. A 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 compiled appropriately, also in several embedded devices.

So, the underlying Operating Systems actually would then distinguish between the various Hardware, that is present and with then, ensure that printf would be executed appropriately depending on the Hardware. So, what we achieve with Portability is that essentially Applications will not change even though the underlying Hardware changes.

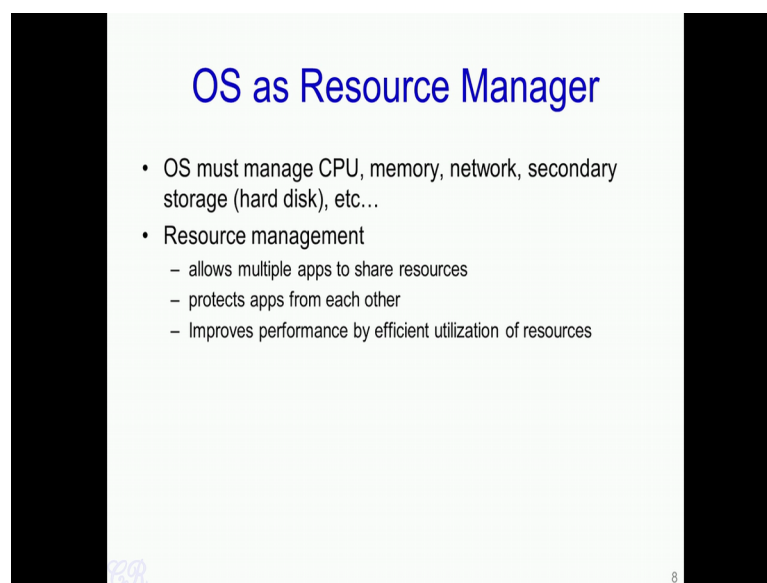
(Refer Slide Time: 10:12)



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 are written and are also executing them or we could be using Skype or a Powerpoint 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 cater to 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)



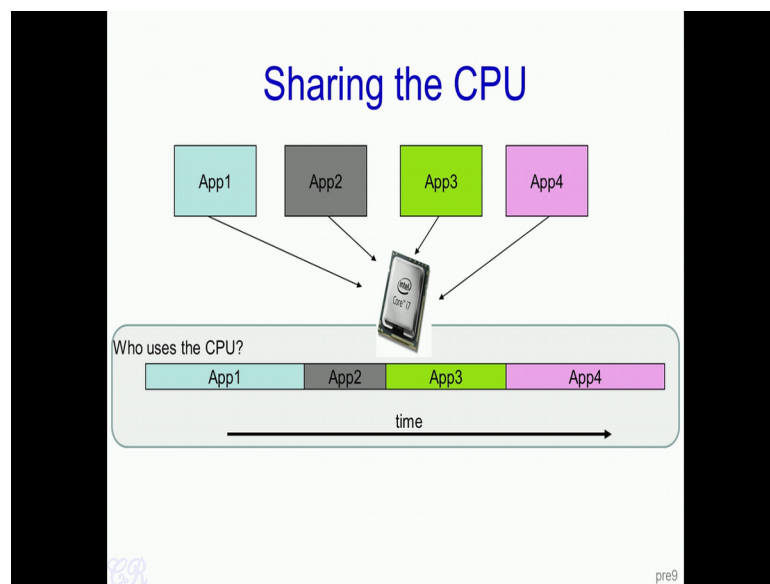
The slide is titled "OS as Resource Manager" in blue text. It contains two main bullet points. The first bullet point states that the OS must manage CPU, memory, network, secondary storage (hard disk), etc... The second bullet point is "Resource management", which has three sub-bullets: "allows multiple apps to share resources", "protects apps from each other", and "Improves performance by efficient utilization of resources". The slide is framed by two black vertical bars on the left and right. In the bottom left corner, there is a small logo that looks like "BR". In the bottom right corner, there is a small number "8".

- OS must manage CPU, memory, network, secondary storage (hard disk), etc...
- Resource management
  - allows multiple apps to share resources
  - protects apps from each other
  - Improves performance by efficient utilization of resources

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. So, 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 is such that every component in the system is adequately utilized.

(Refer Slide Time: 12:35)



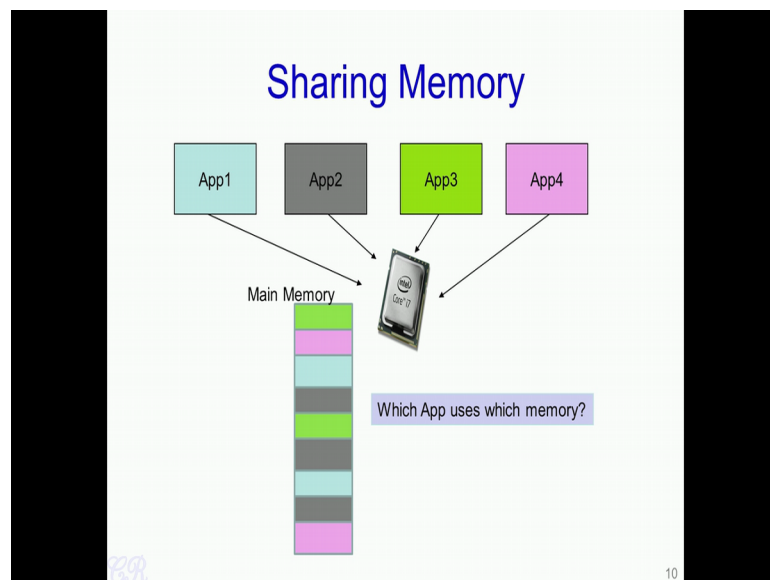
To take an example let us consider the CPU. So, systems typically would have one or two CPUs and multiple applications executing on that CPU. So, how does the Operating System share the single CPU among multiple applications? So, one way which was typically done in the earlier operating systems around the late 70's and early 80's 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



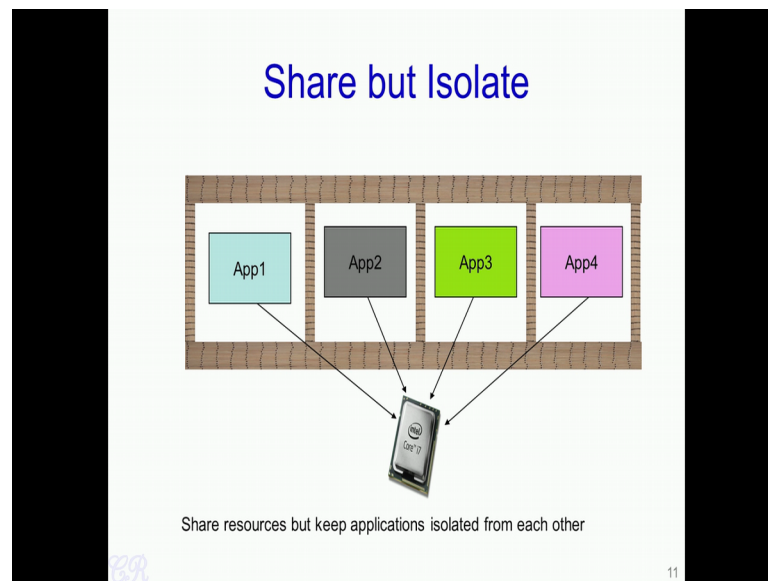
things. So, as we will see in a later video, we will see what are the issues with having Applications execute in this particular way and we will see how modern day Operating Systems manage to utilize this processor in a much more efficient manner.

(Refer Slide Time: 14:13)



An other 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)



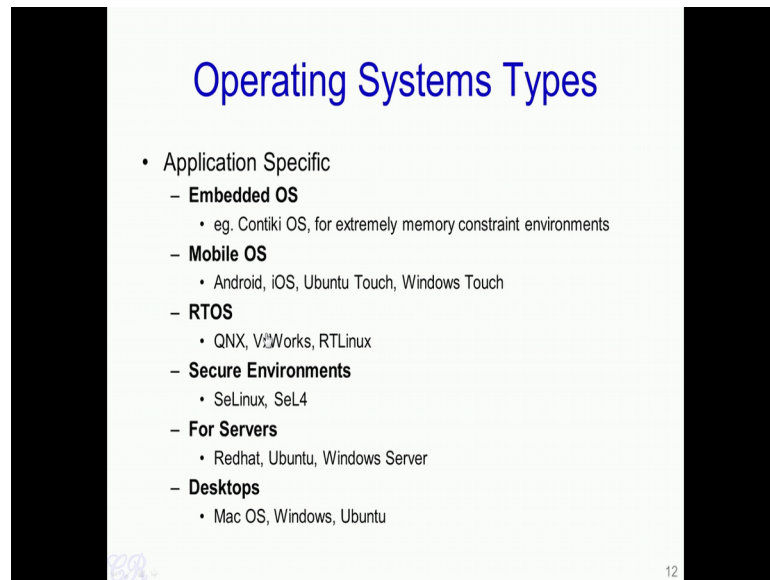
Now, what makes it difficult for the Operating System is that in spite of sharing the limited hardware resources among the various applications which are executing on the computer system, the sharing should be done in such a way so that applications are isolated from each other. So, each application the OS should ensure runs in a 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 a 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 a Gaming software and let us assume that it is having a virus that is it is a 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 steal 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 are ubiquitous.. Almost every smart device that we use today has some form of Operating System present in it. So, this particular slide (mentioned above) shows the various classifications of operating systems depending on the type of application they are intended for. Needless 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, iOS, 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 kept minimum. So 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's like the once we mentioned over here has quite similarities in this aspect with the embedded operating systems like the Contiki OS.

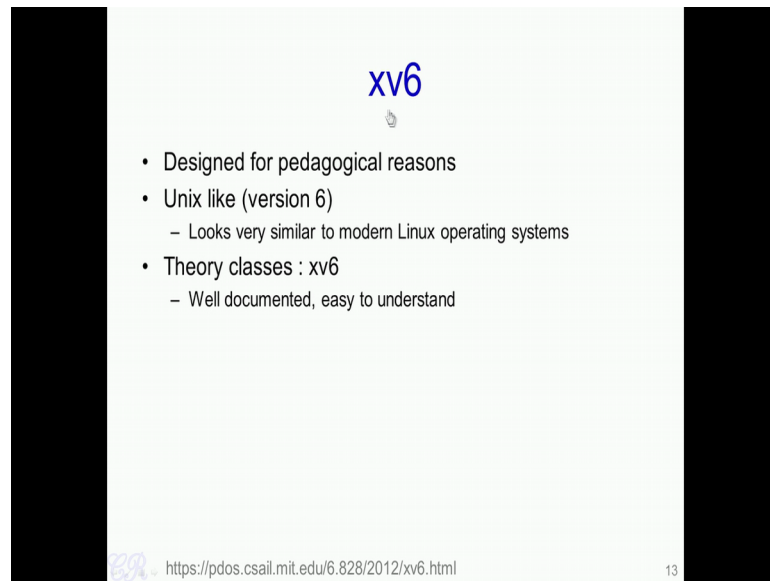
However, mobile operating systems unlike the Embedded OS's also need to take care of certain special devices. For instance the monitors or the LCD screens and key pads. In other words, the mobile OS also has user interaction which typically is not present in the Embedded OS.

A third type of operating system is the RTOS or Real Time Operating Systems. So, examples of these are QNX, VxWorks 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 plants, a delay in doing a particular operation by the computer system would be catastrophic. So, these RTOS's are designed in such a way that every critical operation on the system is guaranteed to complete within the specified time.

Another classification of operating systems are those used in Secure Environments. So, examples of these are SeLinux and 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)



xv6

- Designed for pedagogical reasons
- Unix like (version 6)
  - Looks very similar to modern Linux operating systems
- Theory classes : xv6
  - Well documented, easy to understand

<https://pdos.csail.mit.edu/6.828/2012/xv6.html>

13

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 is 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.