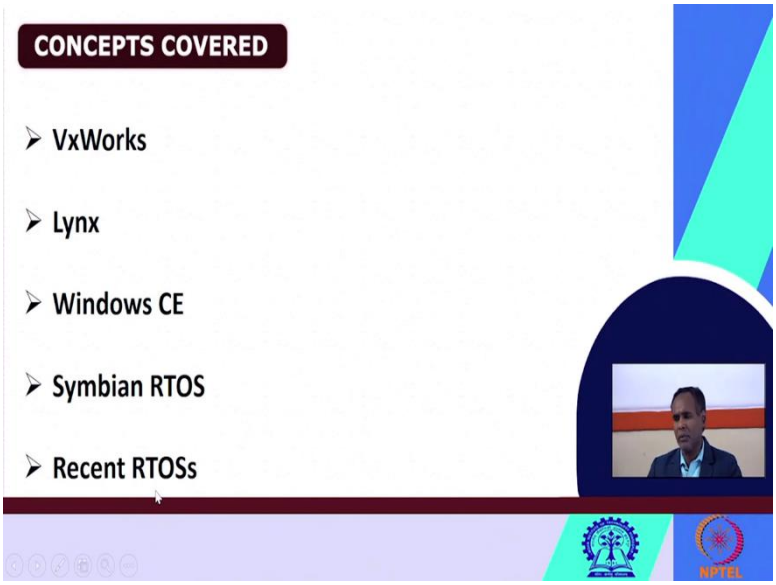
Real Time Systems
Professor Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology Rourkela
Lecture 45
A survey of some contemporary Real-Time Operating Systems (Contd.)

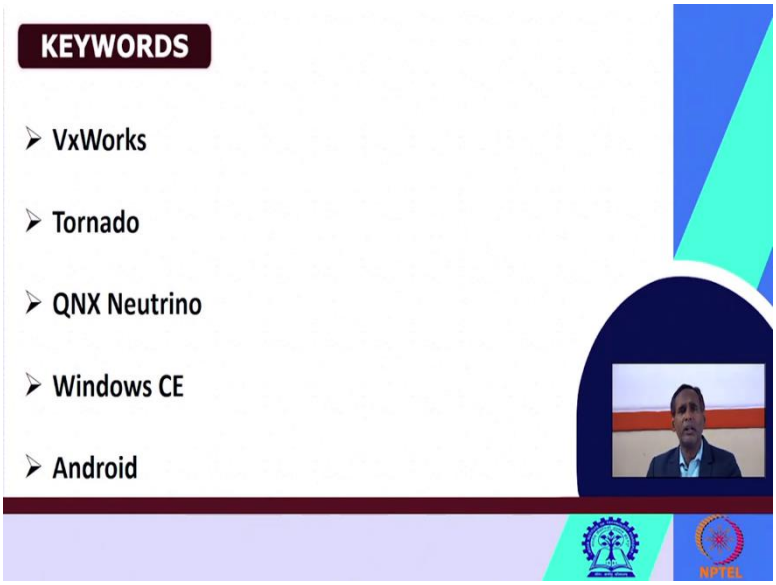
Good afternoon to all of you. Now let us continue where we have left from the last class that is few more contemporary real-time operating system, those are available commercially, we will discuss those things, now.

(Refer Slide Time: 00:29)



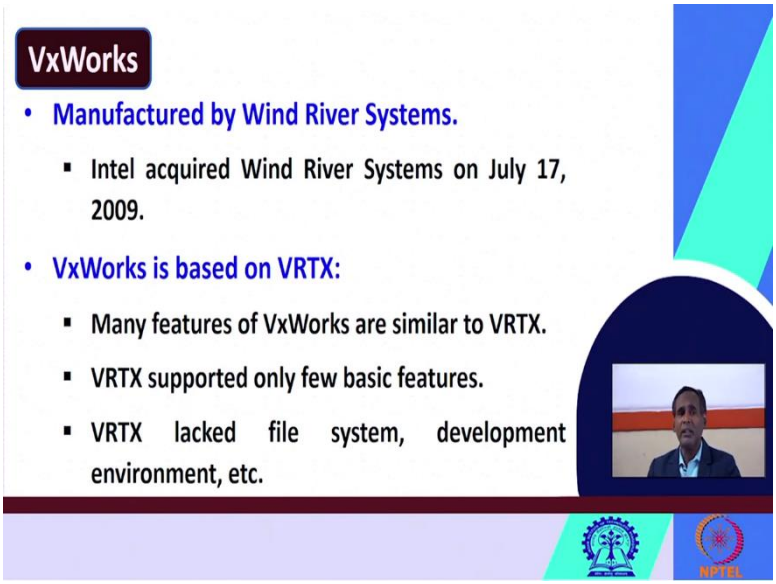
Last class we have already seen about Micro-C Operating System 2, VRTX, etcetera. So, today we will discuss about VxWorks, Lynx, Windows CE, Symbian RTOS, and some other recent modern real-time operating systems we will see.

(Refer Slide Time: 00:48)



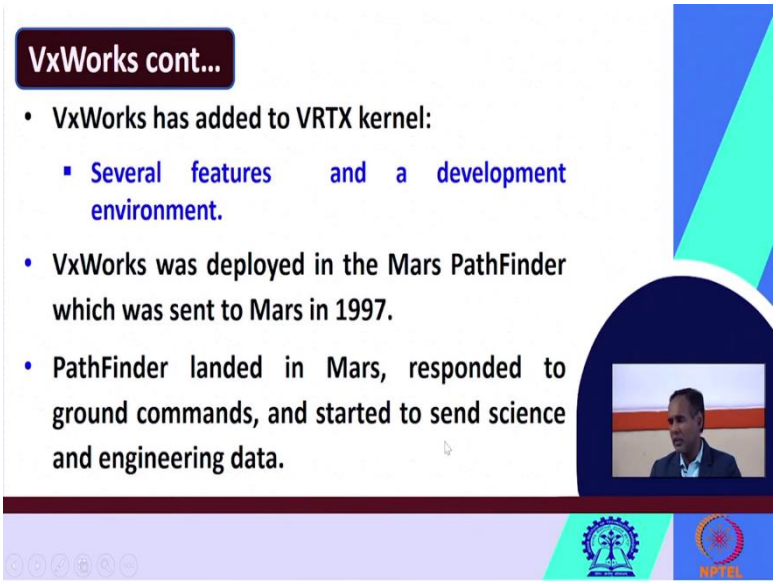
These are the key words we will use like VxWorks, Tornado, QNX Neutrino, Windows CE, and Android.

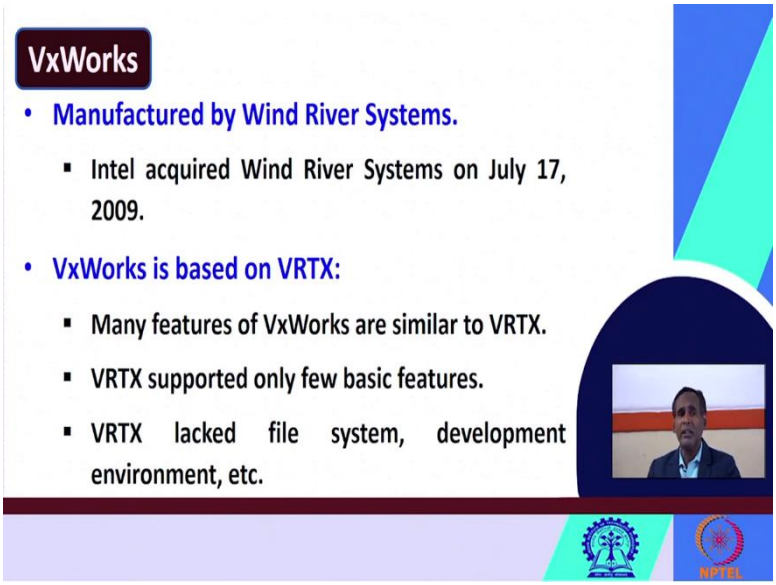
(Refer Slide Time: 00:56)



Let us start with the VxWorks. So, this real-time operating system VxWorks it was manufactured by a company named Wind River Systems. Intel acquired Wind River Systems on July 17, 2009. This VxWorks it is based on VRTX, VRTX already we have discussed in the last class. Many pictures of VxWorks are similar to that of VRTX. The difference is that VRTX supported only few basic features, very few basic features were supported by VRTX. And VRTX does not support like file system, development environment, etcetera. We will see these features are supported in VxWorks.

(Refer Slide Time: 01:40)





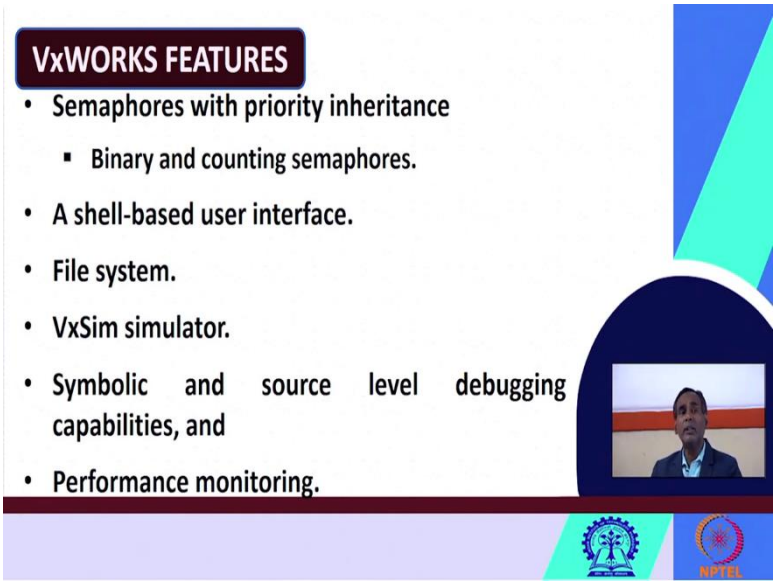
Now, let us see how VxWorks is different from VRTX. So, VxWorks it has added to VRTX kernel some more features and as well as development environment. Please see here I have already told you that VRTX is supported only few basic features, several features we are not supported. Such as lacked file system, it does not support development environment, etcetera. So, these things now they are supported in VxWorks.

So, that is why I say here that VxWorks has added to VRTX kernel. What the following takes, several new features, several advanced features, several additional features and development environment which are not there in VRTX kernel, now they are added to VxWorks.

Now, let us see one of the applications of VxWorks where it was used. So, VxWorks was deployed in the Mars Pathfinder which was sent to Mars in 1997. This Mars Pathfinder, it

landed in Mars, then it responded to the ground commands and started to send various data, those are related to science and engineering, it started to send the various science and engineering data to us.

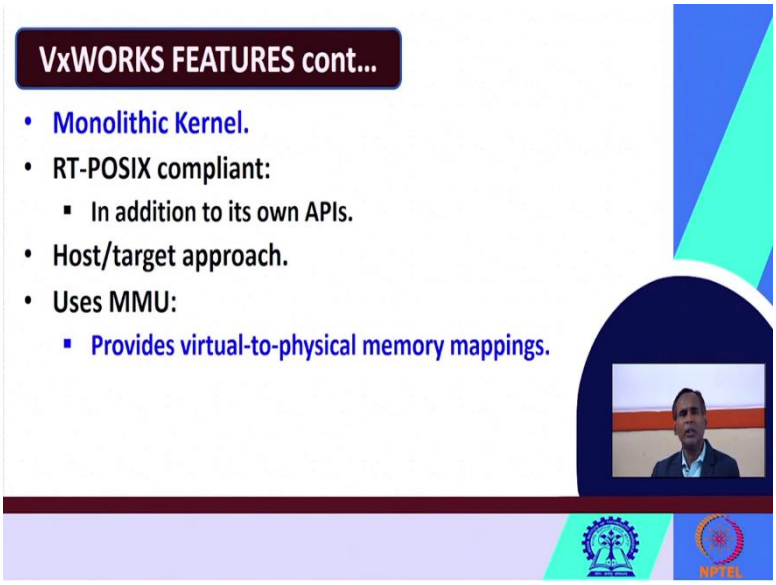
(Refer Slide Time: 02:55)



Now, let us see some of the unique features of VxWorks. Here, the Semaphores, it supports, VxWorks supports Semaphores with a priority inheritance, it uses binary and counting Semaphores. So, it supports Semaphores with priority inheritance, it uses binary and counting Semaphores. So, it uses a shell-based user interface.

This VxWorks it uses a shell-based user interface. It also supports file system, I have already told you VRTX does not support file system, but VxWorks support file systems. It uses a simulator called VxSim. It also provides symbolic and source level debugging capabilities. VxWorks also provides symbolic and source level debugging capabilities, as well as it provides a performance monitoring, it provides a better performance monitoring capability. So, these are some of the unique features of VxWorks.

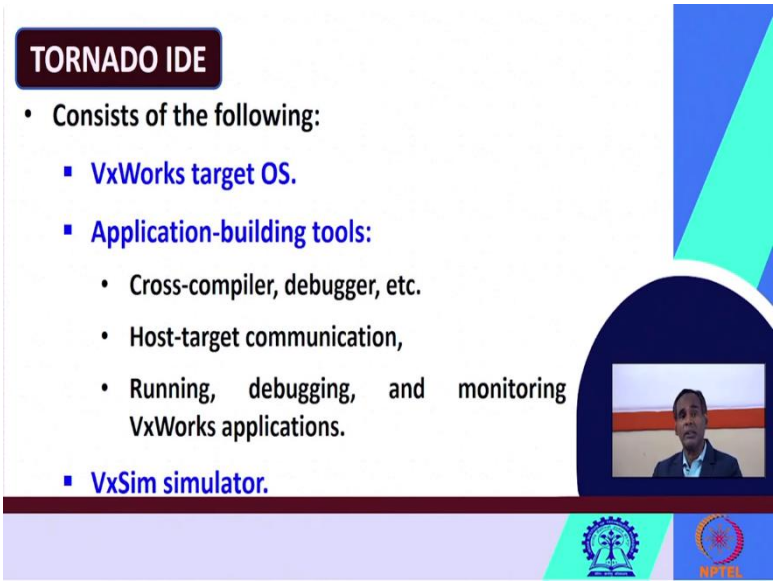
(Refer Slide Time: 03:50)



So then, let us see about its kernel. I have already told you earlier classes there are two types of architectures for the kernel, this monolithic architecture and the microkernel architecture. VxWorks uses monolithic kernel. So, this operating system VxWorks, it is RT-POSIX-compliant. In addition to its own APIs, it is RT-POSIX-compliant. What is RT-POSIX? We have already seen in the last class.

This VxWorks is based on the host target approach. This host target approach also we have discussed in earlier courses while discussing the Unix based real-time operating systems. VxWorks also uses a memory management unit, MMU. So, VxWorks also uses a memory management unit. It provides virtual to physical memory mappings. So, this Vxworks it uses memory management units. It provides virtual to physical memory mappings. These are some of the unique features of the VxWorks.

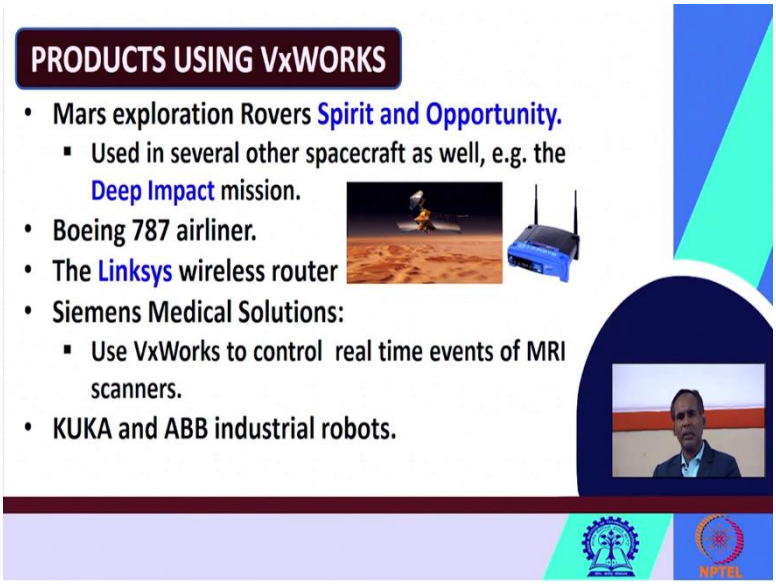
(Refer Slide Time: 04:49)

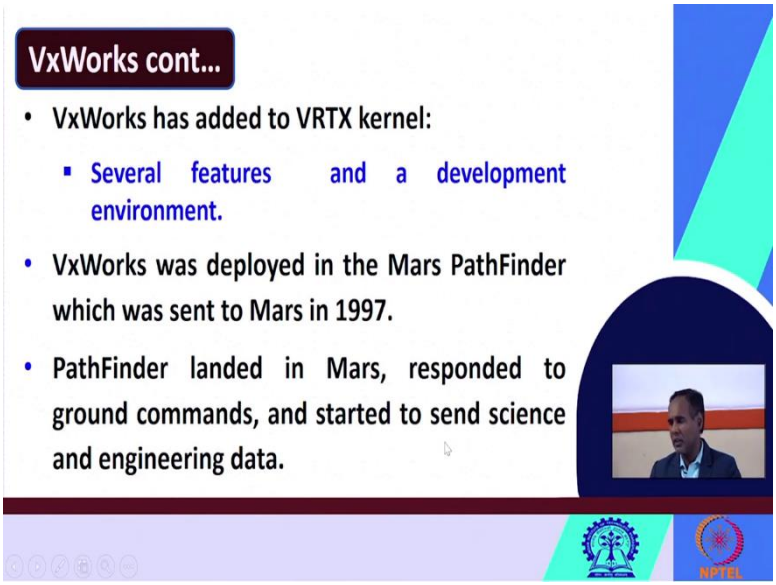


Now, let us see another real-time operating, another important terminology Tornado IDE. This IDE consists of the following, a VxWorks target operating system, some application building tools, and a simulator called VxSim. Now, let us say that this IDE, it contains these things VxWorks circuit operating system, application building tools, and VxSim simulator.

So, this application building tools it contains what? The application building tools include various cross compilers, some debuggers, etcetera. And it uses host target communication. So, it also provides what, it provides running, debugging, and monitoring VxWorks applications, So, these application building tools, they provide capabilities for running, debugging, and monitoring what different or various VxWorks applications. So, these are some of the or this Tornado IDE consists of these following components.

(Refer Slide Time: 06:01)



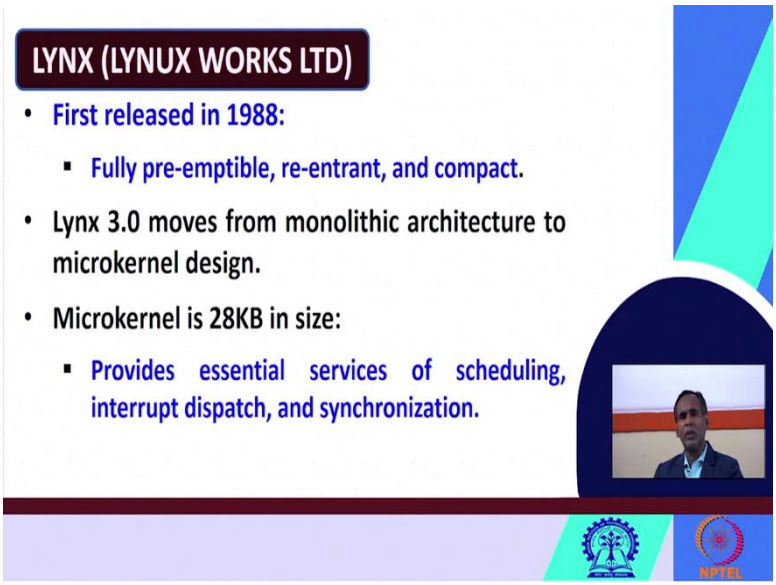


Now, let us see some of the products which use VxWorks real-time operating system. I have already told you this example that your, it was used in this Mars Pathfinder. So, let us see some other products here, this VxWorks were used. Mars exploration Rovers Spirit and Opportunity. The Mars exploration Rovers Spirit and Opportunity, this used in several other spacecraft as well.

So, this VxWorks was used in Mars exploration Rovers Spirit and Opportunity. It was also used in several other spacecraft as well for example, the Deep Impact mission. It is also used in Boeing 787 airliners. It is also used in the Linksys wireless routers. Siemens medical solutions also use VxWorks to control the real-time events of the MRI scanners in these medical applications. So, this VxWorks also it was used in KUKA and ABB industrial robots.

So, these are some of the products where VxWorks was used. These are some of these products where VxWorks is used.

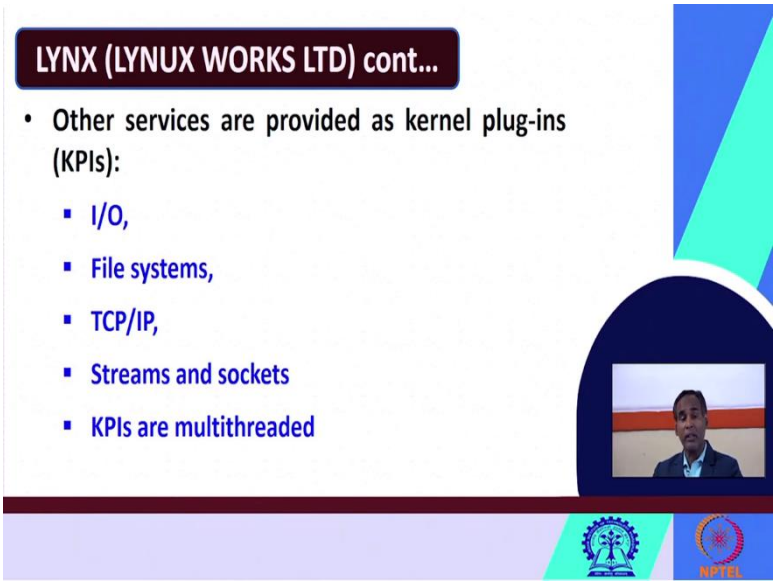
(Refer Slide Time: 07:18)



Now, let us see another real-time operating system called as Lynx. It was first released in the year 1988. So, this real-time operating system it is fully preemptable, it is re-entrant, as well as it is compact. So, earlier the Lynx 3.0 it moved from monolithic architecture to microkernel architecture, that means when on Lynx 3.0 was developed at that time they were using a monolithic architecture and gradually in the advanced versions of Lynx they have switched over from monolithic architecture to microkernel architecture.

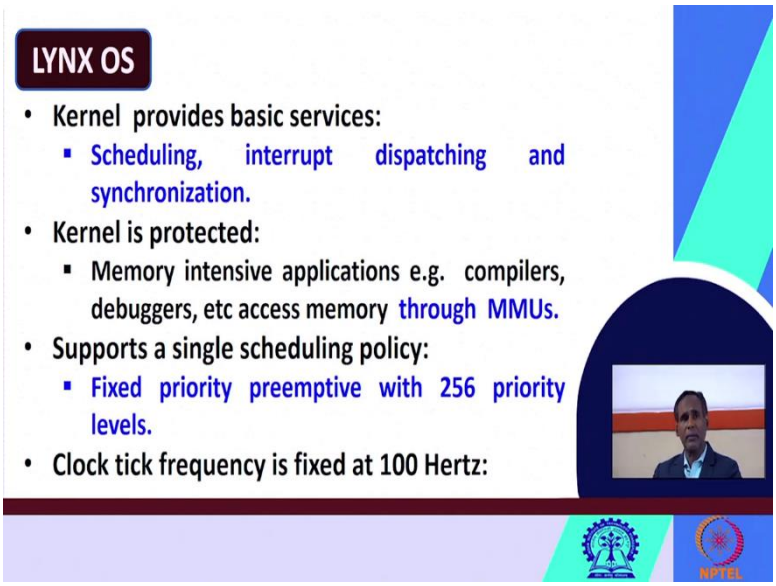
The microkernel size is of 28 kilobytes in case of Lynx. So, microkernel is 28 kilobytes in size. So, this microkernel it provides only the essential services, because you see the size is very, it is quite less. So, this provides only the essential services of scheduling, interrupt dispatching, and synchronization, etcetera. So, only essential services such as scheduling, interrupt dispatch, and synchronization they are performed in this microkernel. So, then, where these other services, how they will be provided, let us see.

(Refer Slide Time: 08:33)



Other services, they are provided as kernel plug-ins. The microkernel provides essential services of scheduling, interrupt dispatch, and synchronization, whereas other services are such as IO related services, handling file systems, TCP IP, handling streams and sockets, KPIs, etcetera. These services are provided as a kernel plug-in, so we call them as KPI. The KPIs here they are used they are normally multi-threaded. The KPIs they are normally multi-threaded.

(Refer Slide Time: 09:06)



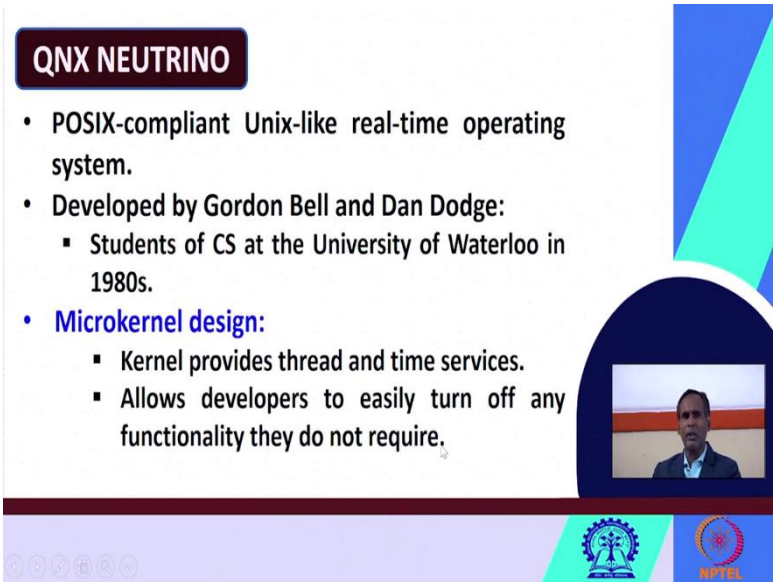
Now, let us see so we are discussing about the Lynx. Now, let us discuss the details or some more detail about Lynx OS. The Lynx operating system kernel provides the basic services, I have already told you those basic services are these things like scheduling, interrupt

dispatching, and synchronization. The kernel is protected. So, memory intensive applications for example, compilers, debuggers, etcetera. They assess the memory throughout through a memory management unit.

So, here, in Lynx operating system the kernel is protected. The memory intensive applications such as compilers, debuggers, etcetera. They access the memory through MMUs, memory management units. So, Lynx operating system supports a single scheduling policy. Please remember, Lynx operating system it supports a single scheduling policy. Fixed priority pre-emptive with 256 priority levels.

So, in this scheduler contains fixed priority pre-emptive, what, maybe tasks with 256 priority levels, there are 256 priority levels maybe from 0 to 255, those priority levels are present in Lynx operating system. In Lynx operating system the clock tick frequency is fixed at 100 hertz. These are some of the features of Lynx operating system.

(Refer Slide Time: 10:32)

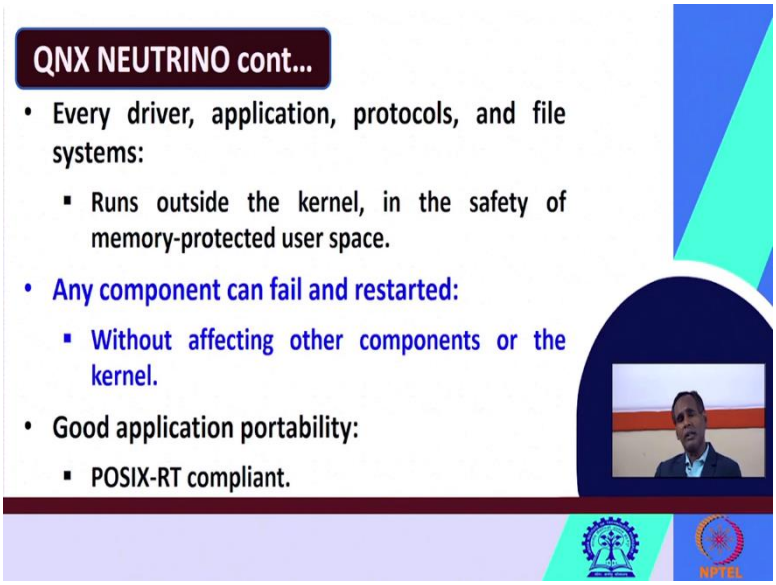


Now, let us quickly go to another real-time operating system that is QNX Neutrino. So, this QNX Neutrino it is a POSIX-compliant Unix-like real-time operating system. So, this real-time system is just like Unix and it is POSIX-compliant. We have already seen what is POSIX-compliant in earlier classes. This operating system was developed by Gordon Bell and Dan Dodge with the students of the CS department at the University of Waterloo in Canada in 1980s.

So, this QNX Neutrino it uses microkernel design. In this microkernel design the kernel provides various thread and the time services. This design allows the developers to easily turn

off any functionality they do not require. So, because this is what to make the size less in order to fit into the memory. So, we do not require all the modules all the functionality always, so whenever we do not require the module or a particular functionality then we can easily turn off. So, this microkernel design allows developers to easily turn off any functionality which they do not require.

(Refer Slide Time: 11:34)

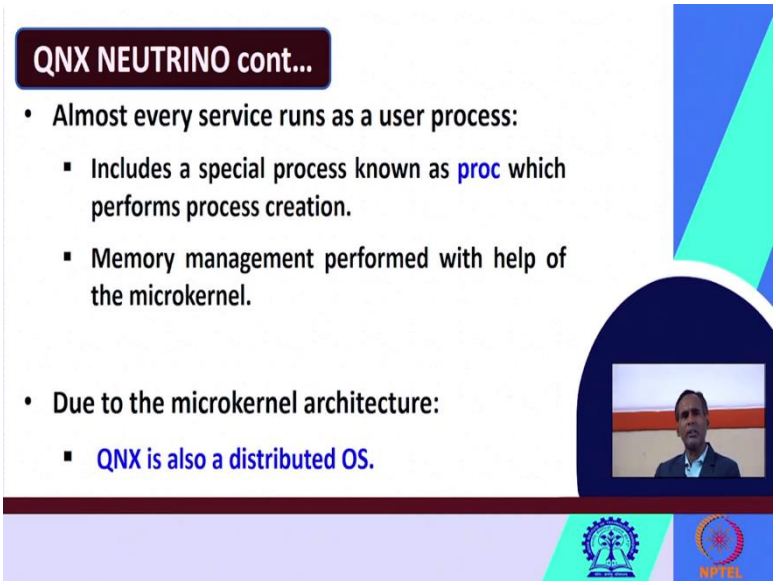


So, every driver, every application, every protocol, and the file system they run outside the kernel. Please remember in QNX Neutrino each driver application protocol and file system they run outside the kernel, in the safety of memory-protected user space. So, in the safety of the memory-protected user space. So, each of the driver application protocol or file system there are on outside the kernel. Any component can fail and restarted. That is possible in QNX Neutrino.

That is special what feature of QNX Neutrino. Any component can fail and restarted without affecting other components or the kernel. So, without affecting any other component or the kernel, any component can fail and restarted. If your component has failed, you can restart it without affecting the other components or the kernel. So, any component can fail and restarted without affecting other components or the kernel.

We have already known POSIX, the objective of the POSIX we have seen that better portability. Since QNX Neutrino is POSIX-RT compliant. So, it provides very good application portability, that means the application written in QNX Neutrino that version provided by one vendor it can easily run in another version or in another system.

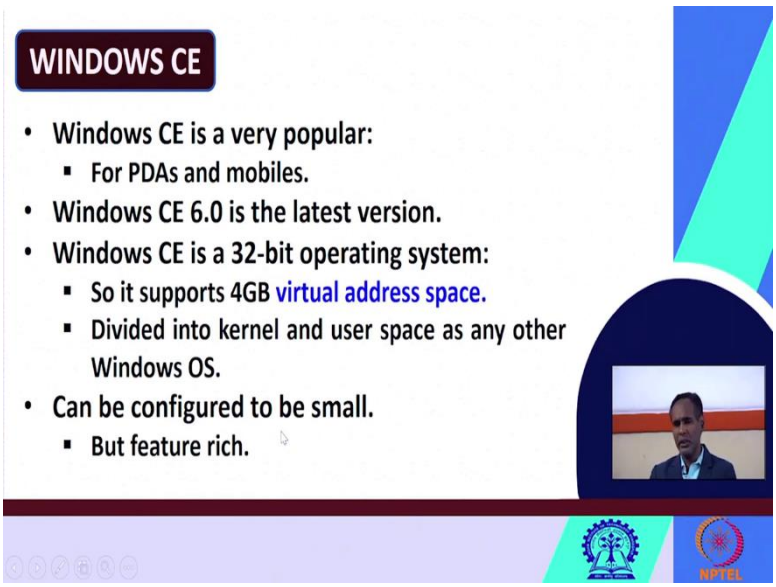
(Refer Slide Time: 13:15)



Almost every service in QNX Neutrino runs as a user process, it includes a special process known as PROC, which performs process creation. So, in QNX every service runs as a user process, it includes a special process known as PROC, which performs process creation. So, the memory management is performed with the help of the microkernel.

So, in QNX Neutrino the memory management is performed with the help of the microkernel. So, due to the use of the microkernel architecture QNX is also considered as a distributed operating system. It is not using a monolithic architecture. It is using a microkernel architecture. So, due to the microkernel architecture QNX is also considered as a distributed operating system. So, this is something about this QNX Neutrino.

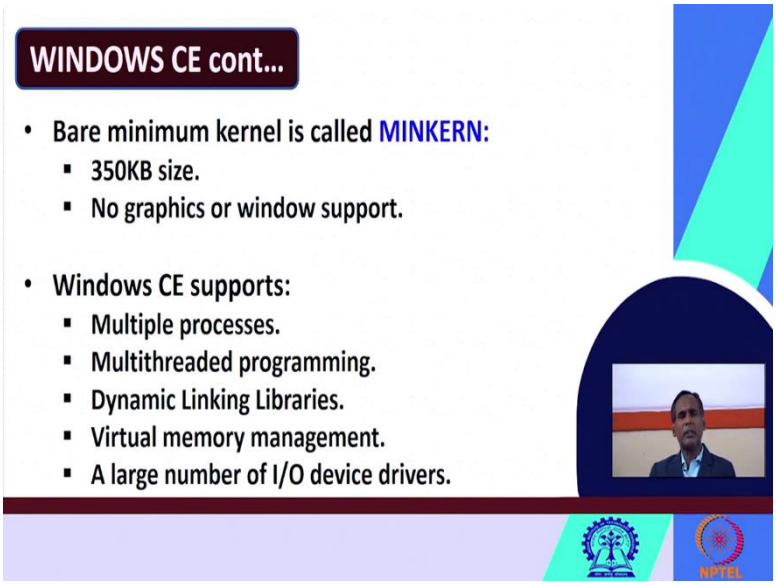
(Refer Slide Time: 14:08)



Now, let us see Windows CE. So, Windows CE you can see that it is a variant of this Windows operating system. Windows CE is a very popular operating system for these PDAs and mobile, PDAs hand held PC. So, Windows CE is a very popular for PDAs and mobile devices. Windows CE 6.0 the latest version. Now, I think you can check right now it is already 6.0 after that 7.0 and 8.0 has come up.

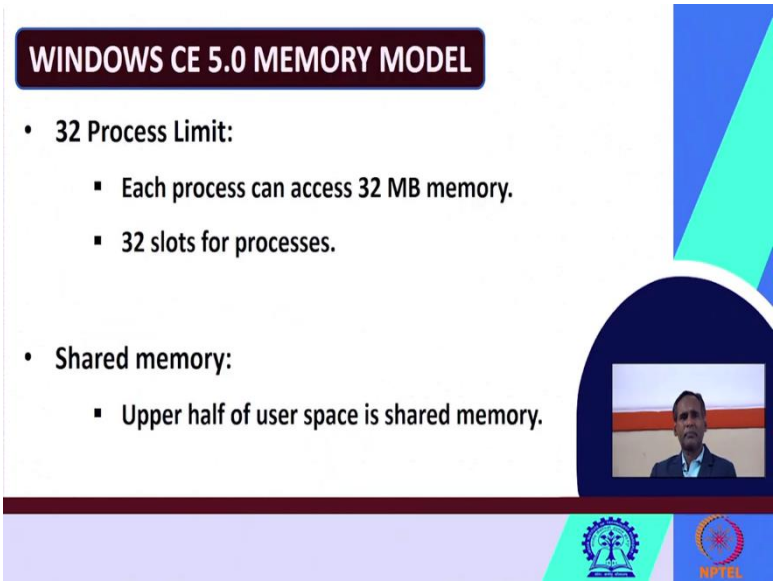
So, Windows CE 8.0, now I think it is the latest version you can check from the net. Windows CE it is a 32-bit operating system. So, it is supports 4GB virtual address space. It is divided into kernel and user space. So, this 4GB virtual address space it is divided into kernel and user space, as in any other Windows operating system. It can be configured to be small, but very feature rich. So, this Windows CE it can be configured to be very small, but it is very feature rich, so many features are provided in Windows CE.

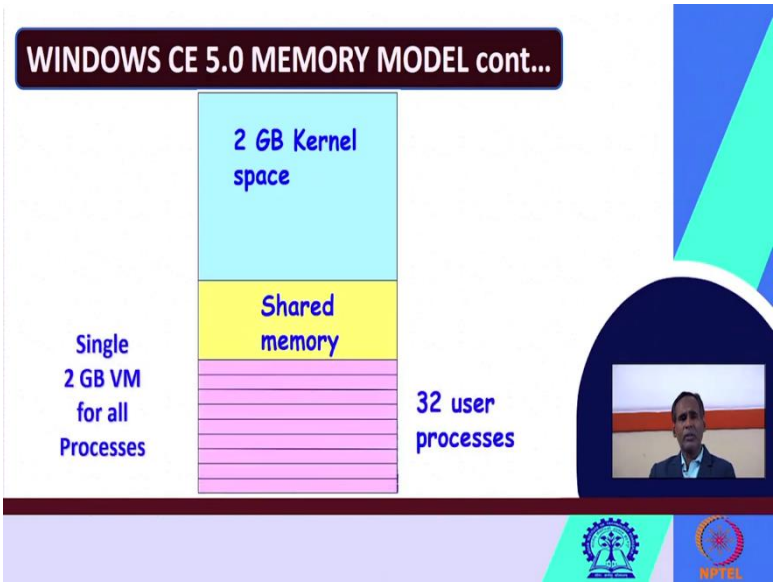
(Refer Slide Time: 15:17)

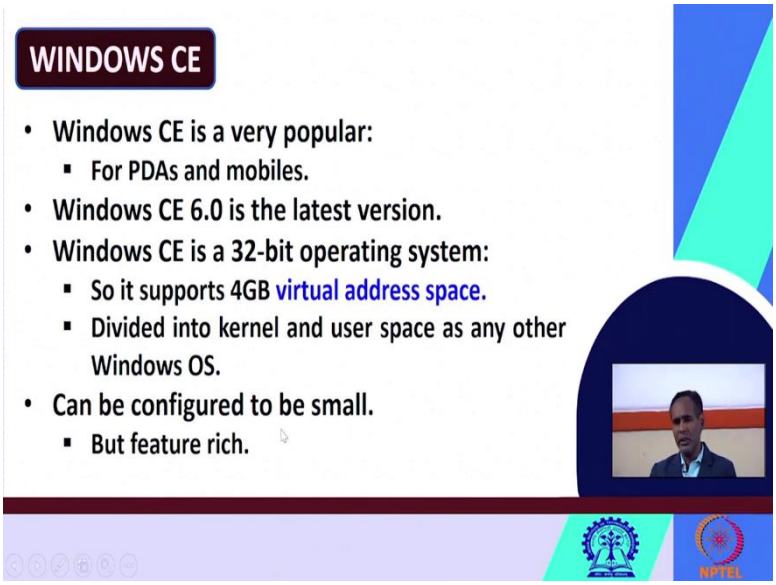


The bare minimum kernel in Windows CE is called as MINKERN. Its size is 350 kilobytes, but it does not support graphics and the windows, because the size is very less so it does not support graphics and windows. Then let us see what Windows CE supports. Windows CE supports the following things. It supports multiple processes. It supports multi-threaded programming. It supports DLL, dynamic linking libraries. It also provides virtual memory management. It also provides or it supports a large number of IO device drivers. So, these features are supported by Windows CE.

(Refer Slide Time: 15:58)





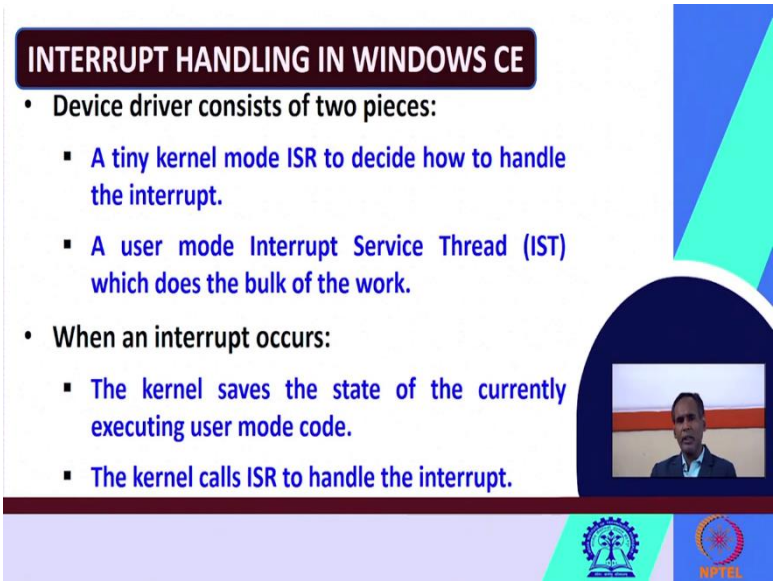


So, let us see this memory model, Windows CE 5.0 memory model. In Windows CE 5.0, how the memory model looks like? What are the special features? It is 32 process limit that means say each process can access 32 MB of memory. And there are 32 slots for the processes. It uses a shared memory, usually the upper half of the user space in the shared memory, I will just show the diagram.

So, it uses shared memory, the upper half of the user space is known as the shared memory, you can see the diagrams that look like this. I already told you here it supports how many that 4GB virtual address space. Here I have already told you, the upper half of the user space is known as this shared memory, this is the upper half of this, you can see this portion in the user space.

So, this upper half in this shared memory and on top of this 2GB kernel space is there. So, there are 32 user processors are there. So, it is having single 2GB virtual memory for all the processes. So, this is how the Windows CE 5.0 memory model looks like.

(Refer Slide Time: 17:13)

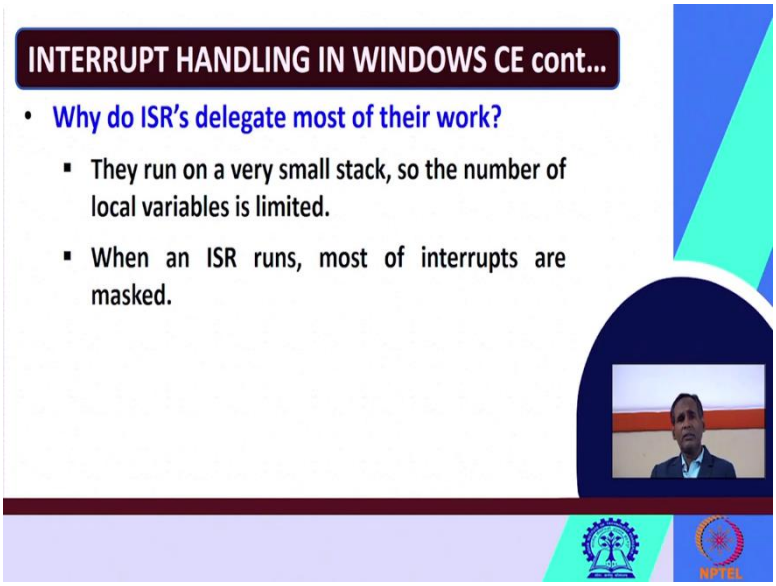


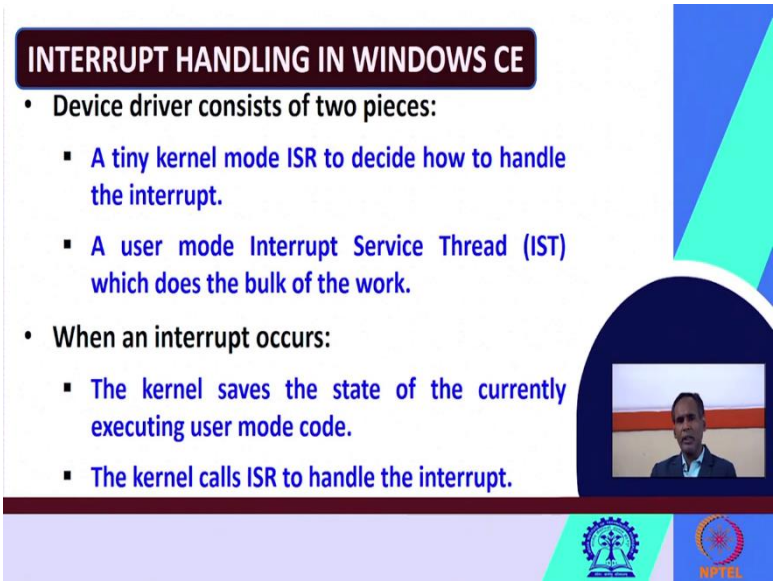
Now, let us see how interrupt handling occurs in Windows CE. So, the device driver in Windows CE consists of two pieces. A tiny kernel mode ISR to decide how to handle the interrupt. And a user mode Interrupt Service Thread, which does the bulk of the work. So, in case Windows CE, the device driver consists of two pieces. First one is it a tiny kernel mode ISR, interrupt service routine to decide what, to decide how to handle the interrupt.

Then what is the second piece, it is a user mode Interrupt Service Thread, IST, what does it do, which does the bulk of the work. Now, let us see how the interrupts they are handled. When an

interrupt occurs the kernel first saves the state of the currently executing user mode code. So, whenever an interrupt occurs first the kernel saves the state of the currently executing user mode code then the kernel calls the respective ISR, interrupt service routine to handle the interrupt. I am repeating again, when an interrupt occurs first the kernel saves the state of the currently executing user mode code, then the kernel calls the respective ISR, interrupt service routine to handle the interrupt.

(Refer Slide Time: 18:34)

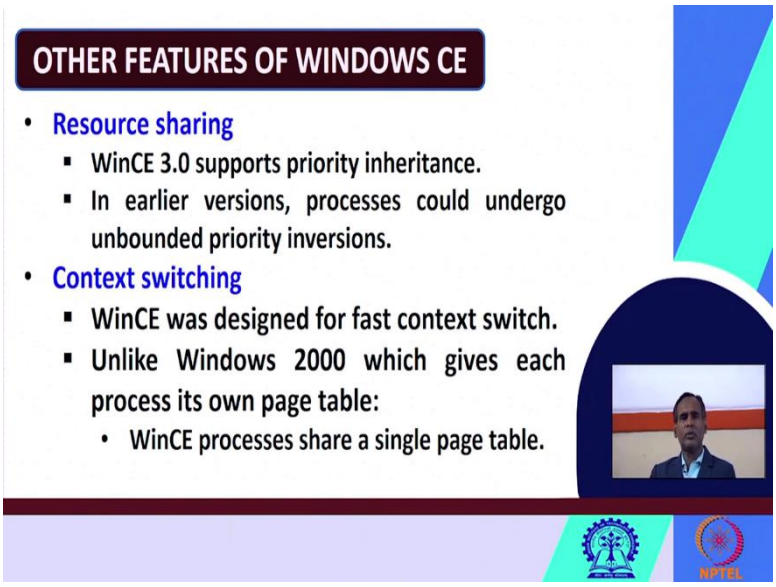




Then why do ISRs delegate most of their work? Here, I have already told you that the kernel calls ISR to handle the interrupt. Now, let us see, why do ISRs delegate most of their work. There are some reasons, why the ISRs they delegate most of their work. They run on a very small stack so the number of local variables is very limited. So, that is why this is one reason.

The ISRs, they run on a very small stack and hence the number of local variables is very less. So, when an ISR runs most of the interrupts are masked. So, when an ISR runs, most of the interrupts they are masked. These are the reasons why the ISRs they delegate most of their work.

(Refer Slide Time: 19:19)



Let us quickly look at the other features of Windows CE. So, one most, one important feature is resource handling. So, Windows CE 3.0 it supports priority inheritance, you have already known what the priority inheritance, different protocols you have shown, PIP, and what, a PIC, etcetera, you know priority inheritance protocol, PCP, priority selling protocol, HLP, etcetera I have known.

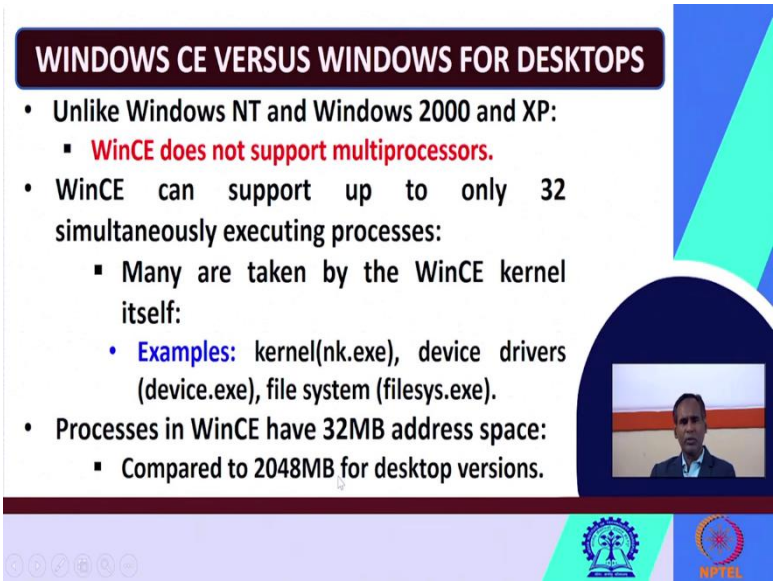
So, Windows 3.0 supports priority inheritance, or supports some protocols to handle priority inheritance. In earlier versions the processes could undergo unbounded priority inversions. So, before Windows 3.0, the earlier versions of Windows CE in those earlier versions of Windows CE, the processes they may undergo unbounded priority inversions but since in Windows CE 3.0 it supports priority inheritance.

So, those priority inversions, those unbounded priority inversions will not arise here. It also another feature is, another feature Windows CE is context switching. So, the Windows CE was designed for a fast context switch. So, normally this Windows CE it was designed for providing fast context switch.

Unlike the normal Windows operating system such as Windows 2000, which gives each process its own page stable, Windows CE processes share, they share a single page table. Please

see, in Windows 2000 what is happening, it gives each process its own page table but unlike that the Windows CE processes they share a single page table. This is one of the differences between Windows 2000 and Windows CE.

(Refer Slide Time: 21:03)

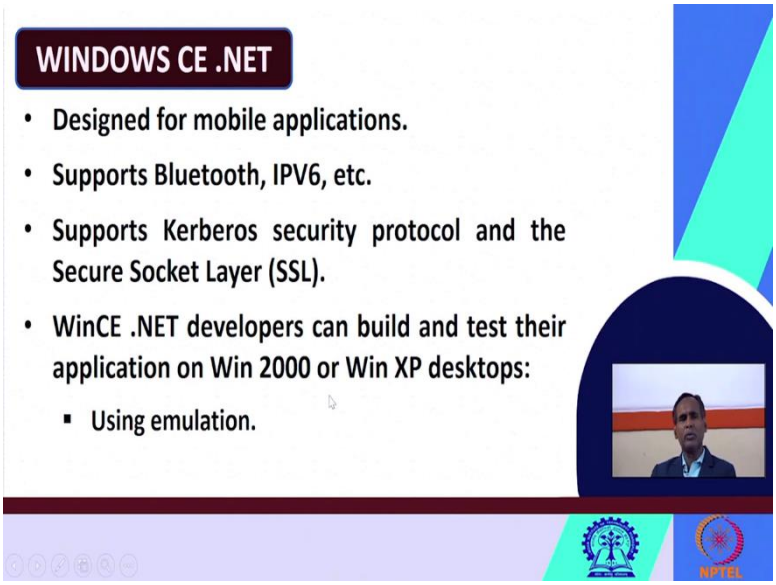


Now, let us quickly see about the Windows CE versus Windows for desktops. Unlike Windows NT and Windows 2000 or Windows XP, Windows CE does not support the multiprocessor please remember, Windows NT and Windows, unlike like this Windows NT and Windows 2000 and Windows XP this Windows CE does not support multi processors. Windows CE can support up to only 32 simultaneously executing processes.

This real-time operating system Windows CE it can support up or it can support up to how many concurrently processes, it can support up to only 32 simultaneously or concurrently my executing processes. Out of these concurrently executing processes are, many of those processes are taken by the Windows CE kernel itself. For example, kernel, device drivers, file systems they are taken by the Windows CE kernel itself.

The processes in Windows CE have 32 MB address space. The processes in Windows CE they have 32 MB address space as compared to 2048 MB for desktop versions. So, normally for the desktop versions the address space size was 2048 MB, but in comparison to that Windows CE have just only 32 MB address space.

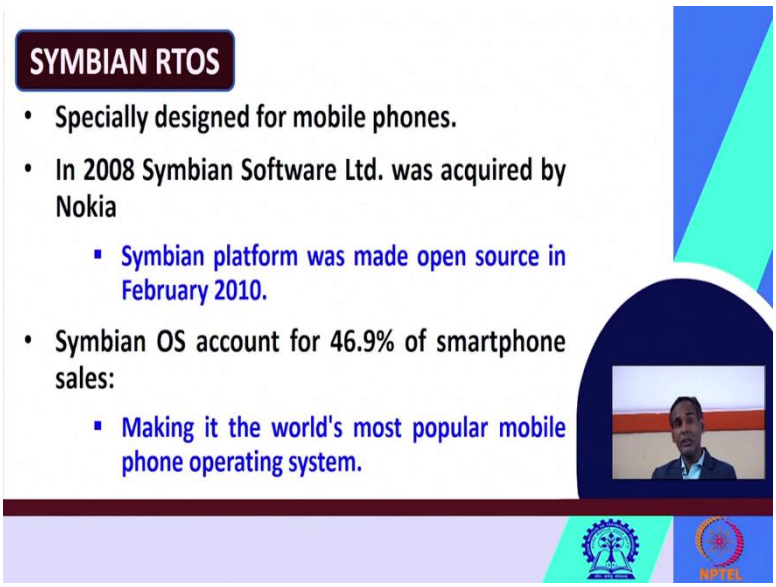
(Refer Slide Time: 22:25)



So, Windows CE dot NET we will see another variant of this Windows CE. It is normally designed for the mobile applications. It supports Bluetooth, IP version 6, etcetera. It also supports Kerberos, those who have studied distributed systems, this is a security protocol in distributed systems Kerberos. So, Windows CE supports Kerberos security protocol and the secure socket layer or SSL.

Windows CE dot NET developers they can build and test their application on Windows 2000 or Windows XP desktops using emulation. So, Windows CE dot NET developers, they can also build and test their applications even on Windows 2000 or Windows XP desktops how by using the concept of emulation. This is a little bit about the Windows CE dot NET.

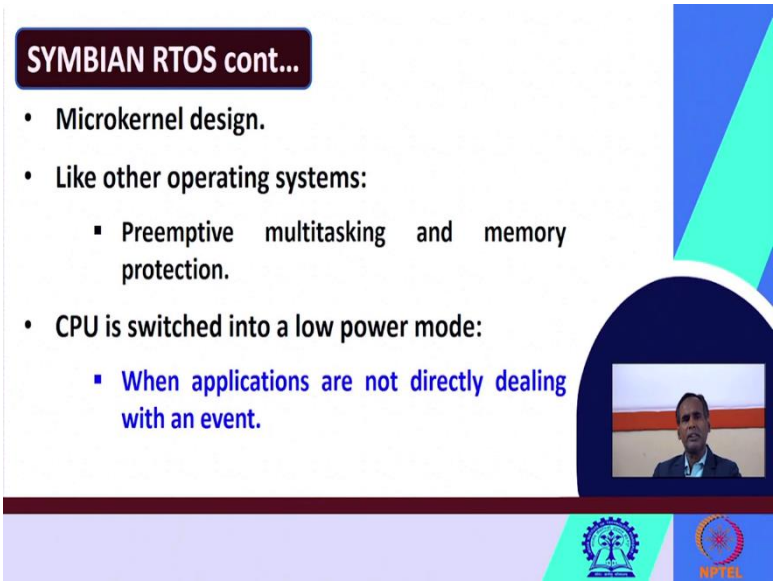
(Refer Slide Time: 23:16)



Now, we will see one more this real-time operating system that is Symbian RTOS, it was specially designed for mobile phones. Actually in 2008, the Symbian Software Limited was acquired by Nokia and this Symbian platform was made open source in February 2010.

Symbian operating system account for on that this Symbian operating system account for 46.9 percent of smartphone sales. Making it the world's most popular mobile phone operating system, is not it? Nowadays many of you are using what these Nokia phones this, in almost all of these smartphones you see they are using the Symbian operating system. So, Symbian operating system counts for 46.9 percent of this smartphone sales, which makes it the world's most popular mobile phone operating system. Many of these mobile phones, many of the users of the mobile phones, they use the Symbian operating system.

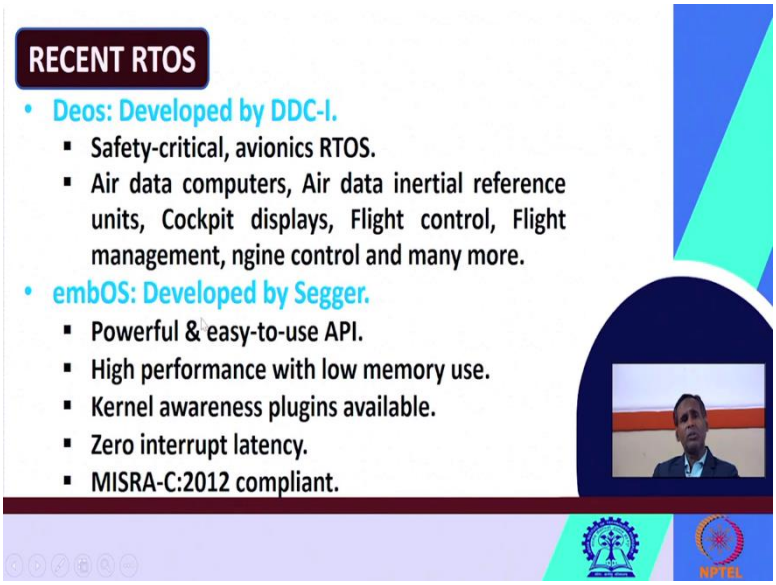
(Refer Slide Time: 24:16)



Now, let us see some of the features of Symbian operating system. It uses microkernel design. We have already seen what is a monolithic design and microkernel design. Like the other operating systems, it also uses pre-emptive multitasking and memory protection, what schemes or memory protection, what techniques.

The CPU is switched into a low power mode; in case of Symbian real-time operating system the CPU is switched into a low power mode. That means when applications are not directly dealing with an event. In case of Symbian real-time operating system, the CPU is switched into very low power mode in order to consume less power when the applications are not directly dealing with an event. So, these are some of the basic concepts of Symbian real-time Operating System.

(Refer Slide Time: 25:00)

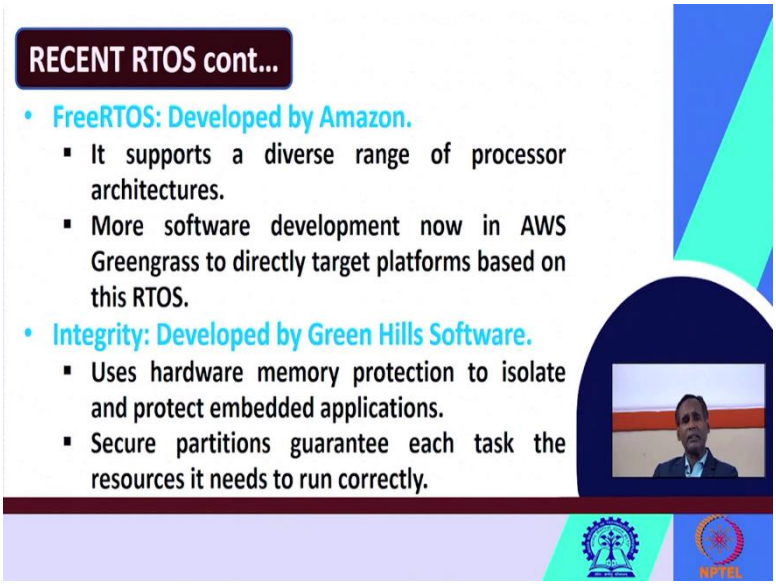


Now, let us quickly look at some of these recent real-time operating system, some of the modern real-time operating systems which are available commercially. The first real-time operating system, the first modern real-time operating system will see as DEOS. So, these are developed by DDC-1. And normally this is used, these are developed for the safety critical applications, avionics application.

So, this is safety critical and avionics real-time operating system. Where it can be used? It can be used in air data computers, air data inertial reference units, cockpit displays, flight control, flight management, engine control, and many more. So, in these cases, this DEOS can be used. Another modern real-time operating system age embOS. So, these are developed by Segger.

This is a very powerful and easy to use API. It shows high performance with low memory use. The kernel awareness plugins are available here. Here zero interrupt latency is there, in case of embOS that is zero latency, zero interrupt latency. So, just like the standard POSIX. So, there is another standard called MISRA-C. So, this real-time operating system embOS, it is MISRA-C 2012 compliant. We will not discuss about this complaint, this POSIX for us this is sufficient.

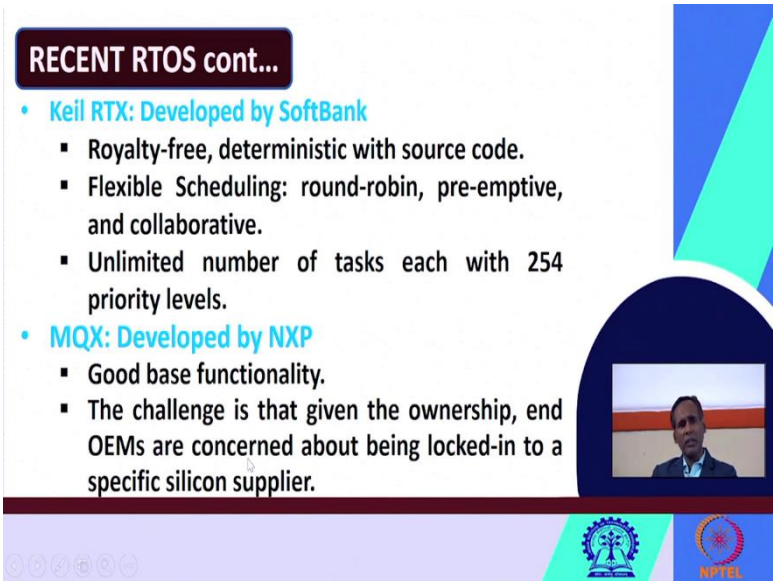
(Refer Slide Time: 26:27)



Another modern real-time operating system is FreeRTOS. These are developed by Amazon. It supports a diverse range of processor architectures. So, this FreeRTOS supports a diverse range of processor architectures, more software development now in AWS. More software development is now in AWS Greengrass to directly target platforms based on this RTOS. More software development is carried out now in AWS Greengrass to directly target the platforms based on this real-time operating system FreeRTOS.

Next type is integrity. This was developed by Green Hills Software. So, it uses hardware memory protection to isolate and protect the embedded applications. So, it uses some secure partitions, the secure partitions guarantee each task the resources needs to run correctly. Now, it uses some secure partitions, this operating system. The secure partitions guarantee that each task or the secure partitions guarantee each task that the resources it needs to run correctly. This is about this operating system integrity.

(Refer Slide Time: 27:35)

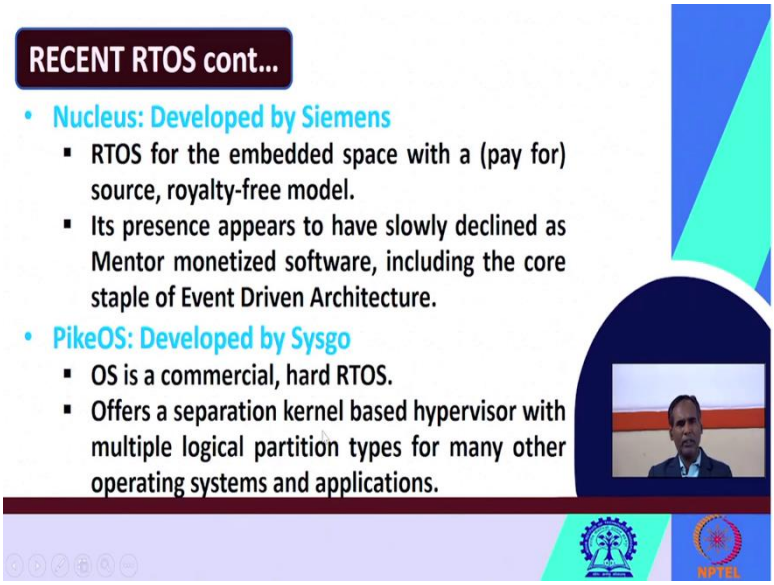


Also, another modern operating, real-time operating system that is Keil RTX. It was developed by SoftBank, it is also a royalty free, it is a royalty free operating system. It is deterministic in nature with the source code then it uses a flexible scheduling. These are real-time operating system uses flexible scheduling such as Round Robin, pre-emptive, and collaborative scheduling techniques. Unlimited number of tasks each with 254 priority levels. So, this real-time operating system supports unlimited number of tasks each with the 254 priority levels.

Then the next real-time operating system the MQX, this was developed by NXP, this real-time system MQX has good base functionality. The challenge in this operating system is that given the ownership the end OEM, I think original equipment manufacturers they are concerned about being locked into a specific silicon supplier.

Let us see what is the challenge in MQX. The challenge in MQX is that given the ownership that the end OEMs, the end Original Equipment Manufacturers they are concerned about being locked into a specific silicon supplier. These are some of the recent or modern real-time operating systems we have discussed. There are several other modern operating, modern real-time operating systems, you can sort them from the net.

(Refer Slide Time: 29:02)

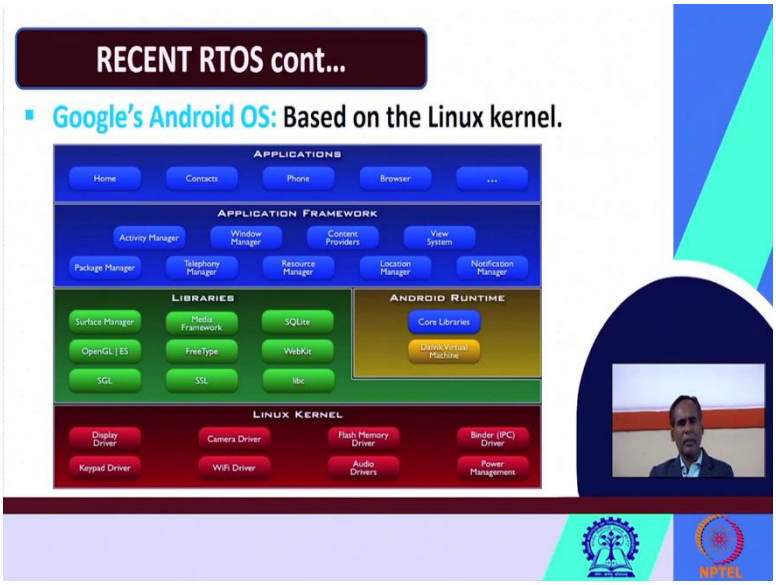


So, one more real time. So, two are there you can see here, one is a Nucleus which are developed by Siemens. So, this is a real-time operating system intended for the embedded space with a source royalty-free model, of course, you may have to pay a little bit, so this is a real-time operating system intended for the embedded space with a source maybe with paying something royalty-free model.

Its presence appears to have slowly declined as a mentor monetized software, including the core staple of event driven architecture. So, the presence of this operating system nucleus appears to have slowly declined as the mentor on monetized software including the core staple of event driven architecture. Then one more recent real-time operating system is PikeOS, this was developed by Sysgo.

This operating system is your commercial hard real-time operating system. This offers a separation, this offers a separation kernel-based hypervisor with the multiple logical partitions, partition types for many other operating systems and applications. So, this real-time operating system PikeOS, it offers a separation kernel based on what, or it provides a hypervisor actually, this real-time operating system offers a separation kernel-based hypervisor with multiple logical partition types for many other operating systems and applications.

(Refer Slide Time: 30:29)



And the last one today we will discuss on this modern operating system is this one. This is about the Google's Android operating system, many of you are using Android operating system in your mobile phone. So, this Android operating system is based on the Linux kernel. Please look at the architecture at the bottom level this Linux kernel is there, you can see there are several drivers, of the display driver, keyboard driver, camera driver, web driver, Wi-Fi driver, then flash memory driver, audio drivers, and then maybe what are some other drivers here you can see.

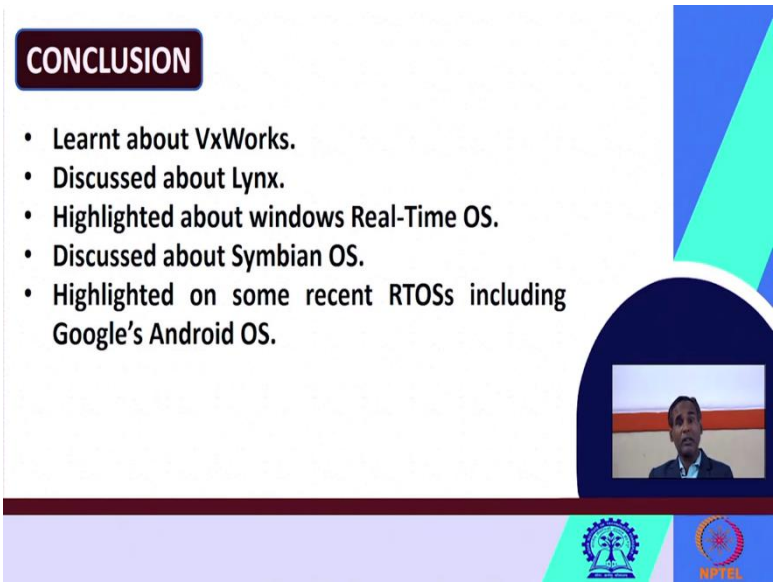
And finally, the Power Management module is also there. On the top there are the various applications are there, such as the home and then the contacts, phone browser and so on, many applications are there on the top layer. Then below the applications the application framework there you, or like various managers are there. So, the activity manager, package manager, technology manager, window manager, etcetera. These are present in the activation manager.

And next layer, various libraries are there, and a sublayer is there, called as Android Runtime. So, various libraries you can see like this, you can say that surface manager and SSL, SGL. And what you can see that Freetype are or something. So, these are the libraries available here. And in the sub layer, Android runtime, we can see the core libraries are there, as well as the Dalvik virtual machine is there.

So, like this. So, this Google's Android operating system looks like, it is based on the Linux kernel. You can see at the bottom layer, the Linux kernel is there and the top-level applications are there and below application, the application framework and libraries are there. There is a what sub layer called Android runtime is also there.

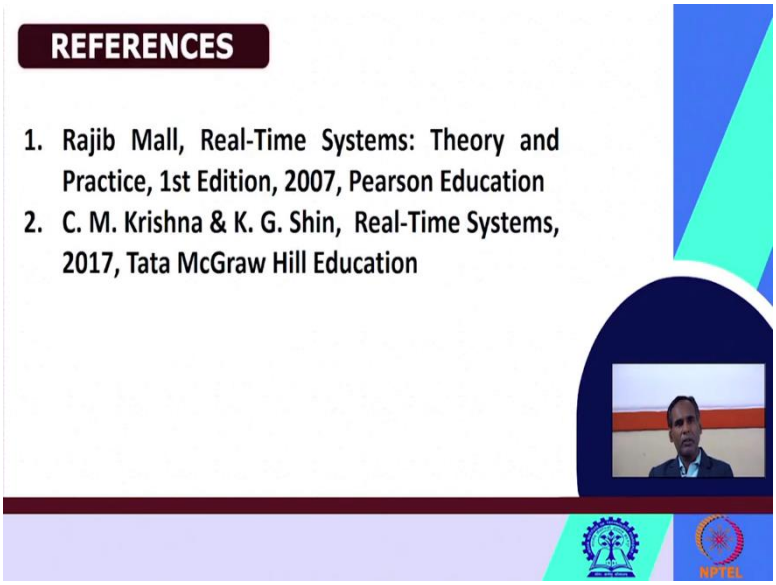
This is how the architecture of the Android operating system looks like. You have already known that Android operating system was developed by Google. This was the architecture of Android operating system. All of you are using in your mobile phones, this shows the popularity of this Android operating system.

(Refer Slide Time: 32:44)



So, today we have discussed about first the operating system VxWorks, then we have discussed about Lynx, then we have discussed about Windows CE, then we have discussed about Symbian operating system. Finally, we have discussed the basics or the fundamentals of several recent or modern real-time operating systems, including the Google's Android operating system.

(Refer Slide Time: 33:05)



We have taken these things from these two books and from some internet sources. Thank you very much.