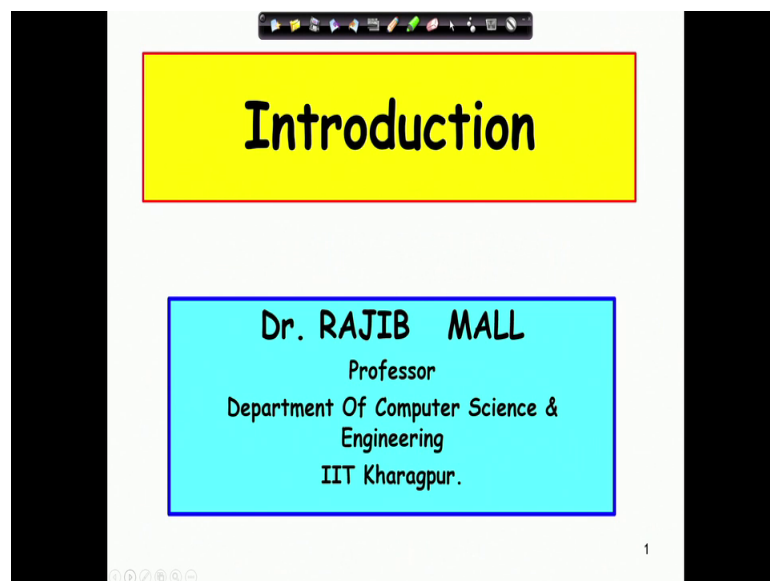


Real Time Operating System
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 01
Introduction

Welcome to this course on Real Time Operating System. Today we will start with some basic introduction and then we will build on this topic.

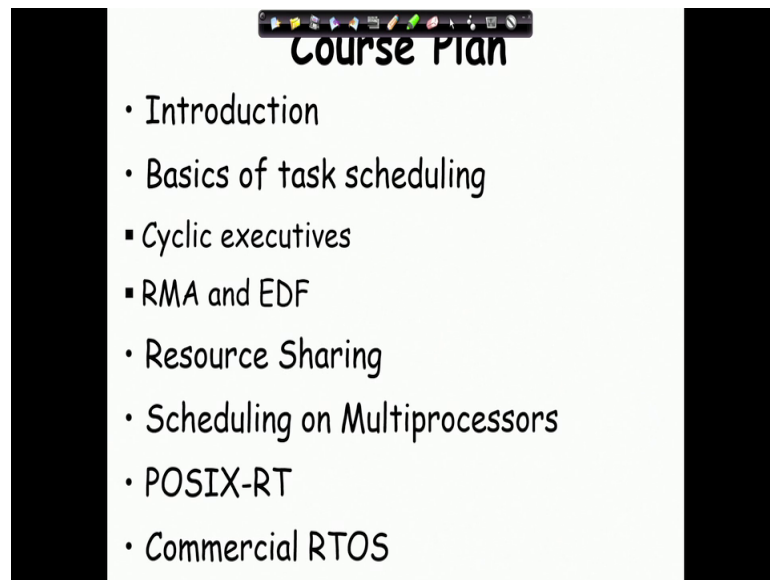
(Refer Slide Time: 00:27)



And by the time we complete course should have good background in various aspects of real time operating systems and then of course, you need to practice. The practice using a actual real operating system is not part of this course, it just gives an overview background to get started with real time operating system uses. So, let us get started with very basic introduction.

So, the first question that we need, somebody would ask is that what is a real time system what is a real time operating system, how is real time operating system different from a traditional operating system, what are the features of this operating system, what are the standards etcetera, what are the commercial operating systems that are available. These are the questions that somebody would have in mind when he or she tries to learn a course on real time operating system. So, let us get started.

(Refer Slide Time: 01:48)

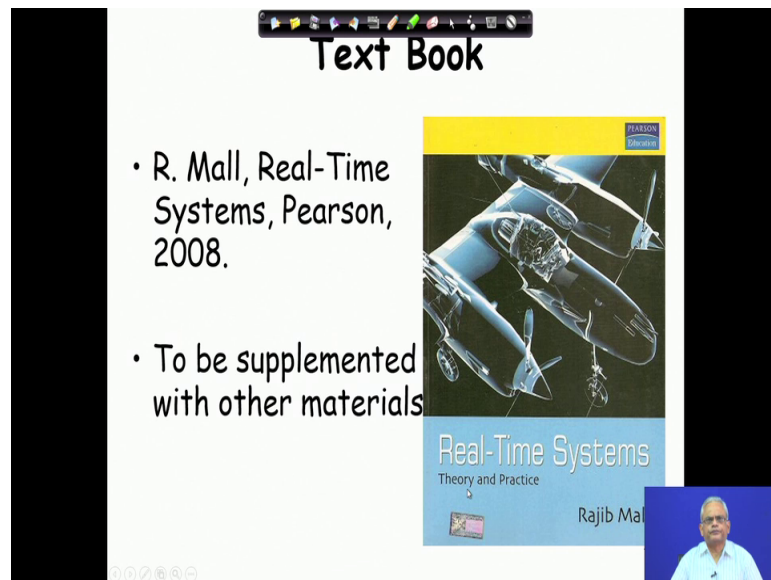


The plan of this course is that this first lecture we will start with some very basic introduction and then we will look some basics of task scheduling actually as a proceed with course we will see that one of the major issues in real time operating system is tasks scheduling.

We will look at different types tasks schedulers that are used in different situations. For example the cyclic executives the rate monotonic algorithm and the earliest deadline first algorithm. And then we will see that how can resource sharing be efficiently achieved in a real time application because we will see that its contrast to an ordinary application where we use semaphores here using a semaphore will lead to some problems, we will analyse the problem and see what are the solution and then we will look at how do we use a real time operating system in a multiprocessor.

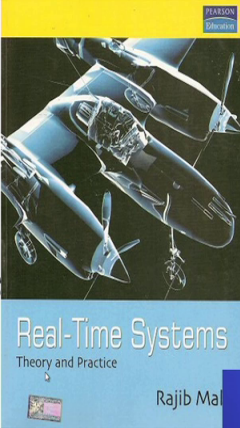
Because nowadays multiprocessors are becoming common place even the desktops embedded devices have multiprocessor and then we will look at the Posix real time standard and then we will look at some of the commercial real time operating system.

(Refer Slide Time: 03:22)



Text Book

- R. Mall, Real-Time Systems, Pearson, 2008.
- To be supplemented with other materials

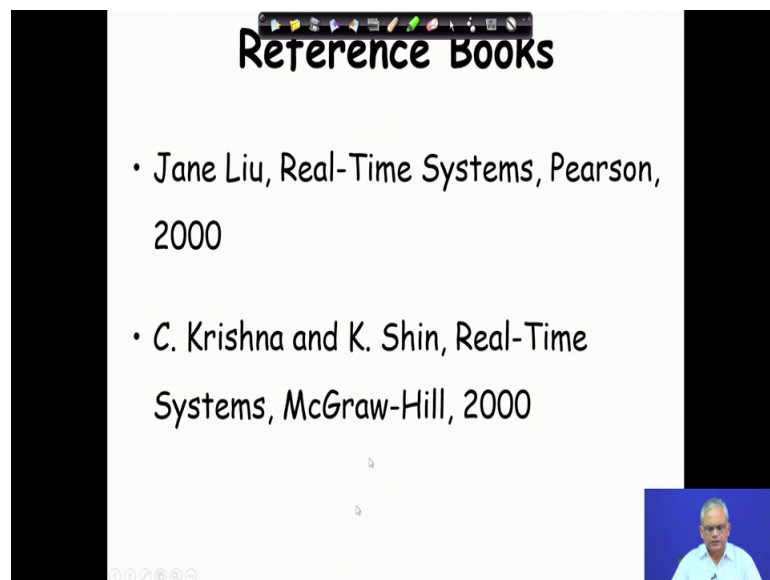


Real-Time Systems
Theory and Practice
Rajib Mall

A small video inset of the presenter is visible in the bottom right corner of the slide.

So, mainly we will use a text book written by me sometime back. So, that is real time systems theory and practice by Pearson press.

(Refer Slide Time: 03:37)



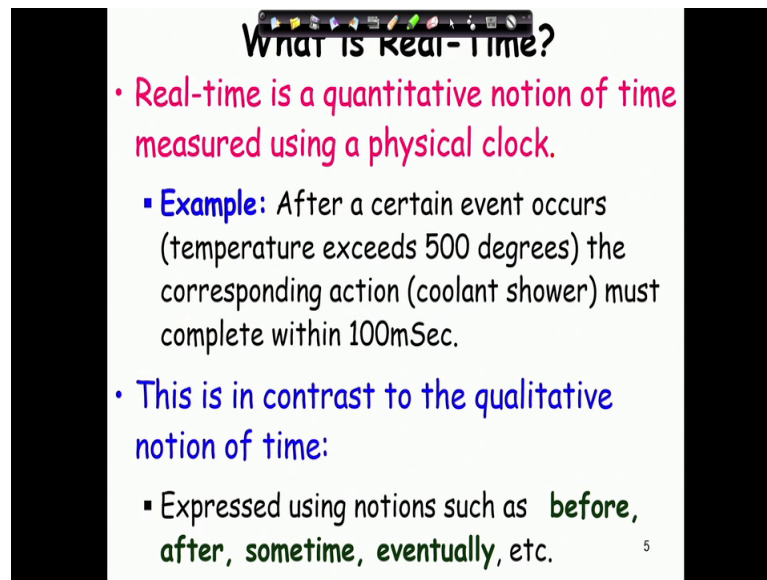
Reference Books

- Jane Liu, Real-Time Systems, Pearson, 2000
- C. Krishna and K. Shin, Real-Time Systems, McGraw-Hill, 2000

A small video inset of the presenter is visible in the bottom right corner of the slide.

And we will supplement this with Jane Liu real time systems, again Pearson and then we will also refer to Krishna and shin real time systems McGraw-Hill.

(Refer Slide Time: 03:53)



What is Real-Time?

- Real-time is a quantitative notion of time measured using a physical clock.
 - **Example:** After a certain event occurs (temperature exceeds 500 degrees) the corresponding action (coolant shower) must complete within 100mSec.
- This is in contrast to the qualitative notion of time:
 - Expressed using notions such as **before, after, sometime, eventually, etc.**

5

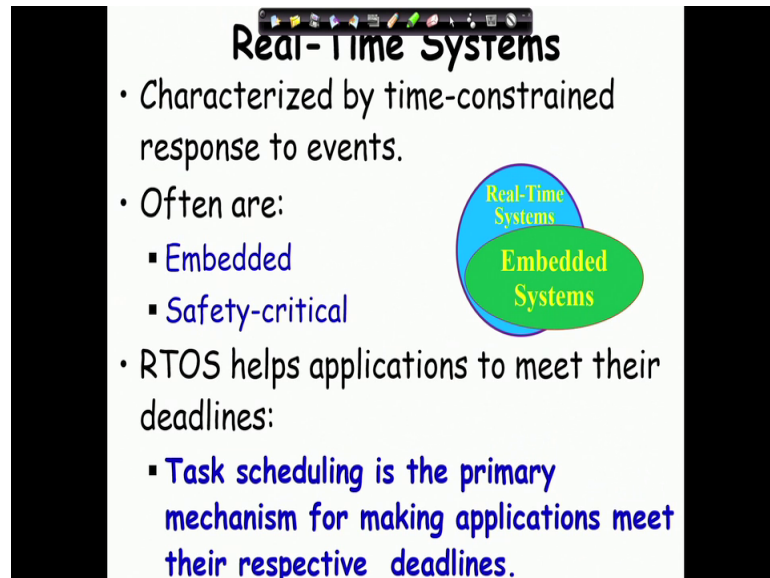
So, the first question is that what is real time? Actually in the simplest term we can say that real time is a quantitative notion of time measured using a physical clock, but then we will have the question that then this is time, so how is it different from other systems they also deal with time. But then here we measure the time explicitly using a physical clock in a real time situation and then if the time bounds are not meant we say that the system has failed.

Just to give an example of a real time system let us say that in a chemical reactor if the temperature exceeds 500 degrees then the coolant shower must be turned down within 100 milliseconds. So, here once the temperature exceeds 500 degrees then we need to set a physical clock and then check whether the coolant shower is actually getting on within 100 millisecond otherwise the system would be considered as failed. In contrast in a traditional system we have the notion of a qualitative notion of time.

In a non-real time system we use terms like before after sometime eventually for example, we might say that in a library before a book can be issued by a member the book records must have been created, the book must have been procured and entered in the systems library. So, just ark here we are just saying that before the member can issue the book the book must have been entered into the system. But that when exactly it would have happened entered, I am not specifying anything it could have been entered the previous day previous hour or may be several years back. So, in the traditional

systems we do not use explicit notion of time we just use a qualitative notion of time like before after sometime etcetera.

(Refer Slide Time: 06:45)



Real-Time Systems

- Characterized by time-constrained response to events.
- Often are:
 - Embedded
 - Safety-critical
- RTOS helps applications to meet their deadlines:
 - **Task scheduling is the primary mechanism for making applications meet their respective deadlines.**

So, in every real time system we have events occurring and then the action must be taken before sometime. So, that we call as the time constrained response to the events and often these systems are embed and safety critical. So, we will see what is an embedded system and what is a safety critical system.

The embedded systems are the ones where the processor the computer the software is part of the system and it controls the system. So, the computer system is embedded within the physical system. There are many examples of an embedded system hundreds of thousands of example for example, let us say a robot or let us say a toy, electronic toy or may be let us say a nuclear reactor or may be let us say a in a automobile there are embedded processors which control the fuel injection or let us say the environmental control like temperature etcetera.

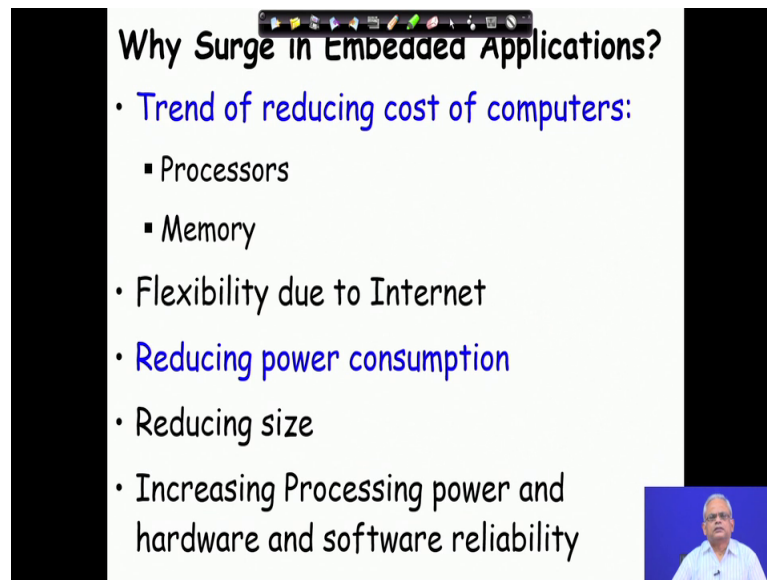
So, there are numerous embedded computers around us and what we want to really say is that the embedded systems majority of them are actually real time systems, but then there are real time systems which may not be actually embedded. For example, let us say a air traffic controller in an airport it reacts to a aircraft coming in and that is not really an embedded system.

So, we have majority of the real time systems or embedded systems, but not all embedded systems are real time systems as shown in this diagram and also there are some real time systems which are not embedded. And many of these are safety critical in the sense that if there is a failure in the system it can cause financial or physical damage. So, may be life property etcetera, so that is a safety critical system I just giving an example of a nuclear reactor. So, if there is a failure it can cause severe disasters loss of life property and so on.

So, the next question is that what is the role of the real time operating systems in these real time systems. Actually the real time operating systems the primary purpose is to help the tasks meet their deadlines there are other features of the real time operating systems as you will see, but primary feature of a real time operating system is that it lets real time applications meet their deadlines, but then the question is that how does a real time operating system help the tasks to meet their deadline.

The answer is that the real time operating systems have a tasks scheduler and it is the tasks scheduler which is the primary mechanism used by the real time operating systems to meet the application deadlines. We will see that how does the tasks scheduler ensures that the tasks meet the deadlines that is one of the primary purpose of this course that what are the different techniques it applies and the programmer says that these are my critical tasks and meeting the deadline of these tasks are crucial and gives the tasks description in the program. And then on the real time operating system takes care to make the application to meet its deadline as various events keep on occurring.

(Refer Slide Time: 11:29)



Why Surge in Embedded Applications?

- Trend of reducing cost of computers:
 - Processors
 - Memory
- Flexibility due to Internet
- Reducing power consumption
- Reducing size
- Increasing Processing power and hardware and software reliability

Video inset: A man in a white shirt speaking.

For last two decades or so, the embedded applications have really increased great extent now the embedded applications vastly outnumbered the traditional computers. For example, in a household we might be possessing just one computer or there may not be any computer, but then almost every household deals with dozens of embedded systems for example a microwave oven, for example a electronic toy, for example a telephone receiver another example may be television controller remote controller etcetera. So, there are many many examples.

So, then why suddenly in last 20 years there is such a large increase in the real time applications. The first thing is or the most important thing is that the prices of computers have really fallen the processors memory have become very cheap. The internet come in and there is tremendous flexibility for different computers to communicate to each other and also the power consumption of the processors and memory have drastically reduced.

Earlier the computers were using so much power that battery operated computer was far pledged, but now embedded processor may work for months or years on battery. The size of computers and memory have really reduced and that has enabled them to be used in very very small real time applications for example, a computer mouse the processor and the software that is there it is almost invisible that and we may have to really wonder that how come such a small processor and the accompanied software memory and so on is there in a small device like a computer mouse. And then the processing power has

tremendously increased and also these are become very very reliable earlier the hardware and software reliability was questionable often the hardware will shut down the software will encounter bugs, failures and so on, but now they have become extremely reliable for them to be used in many many daily applications.

(Refer Slide Time: 14:43)

Important Characteristics

- An embedded system responds to events.

External Input Event → Process New Data → External Output Event

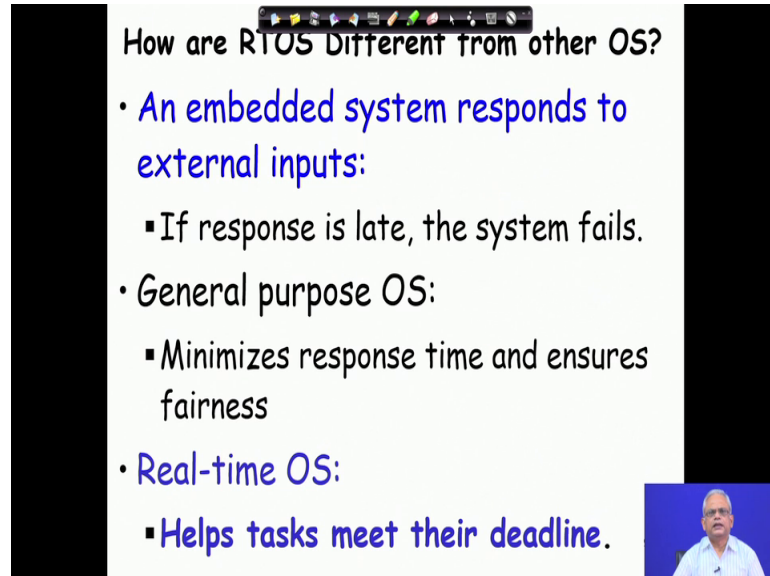
Example: An Automobile airbag system.
When the airbag's motion sensors detect a collision, the system needs to respond by deploying the airbag within 10ms or less.
- or the system fails!

Before we look at the real time operating system let us just spend a minute or to look at the characteristics of these embedded applications because these are one of the major applications areas of real time operating systems. And if we know some of the characteristics of these it becomes easier for you, for us to appreciate the issues that a real time operating system would have to face. So, one thing is that in the embedded system the computer is embedded in the physical system. And there are input external input events it processes and produces output event and that is again fed back to the input processor.

One of the very simple example may be an automobile airbag system. So, whenever there is a automobile crash the motion sensors detect a collision and the system needs to respond by deploying the airbag within ten millisecond or less otherwise we will consider the system to have been failed. So, here just look at it that there is a computer inside this airbag system which is continuously monitoring the events and as soon as the motion sensor indicates that there is a collision the computer acts to deploy the airbag

within 10 millisecond or less. So, there is a time constraint and the action and also the computer is part of the system and it continuously monitors the system.

(Refer Slide Time: 17:00)



How are RTOS Different from other OS?

- An embedded system responds to external inputs:
 - If response is late, the system fails.
- General purpose OS:
 - Minimizes response time and ensures fairness
- Real-time OS:
 - Helps tasks meet their deadline.

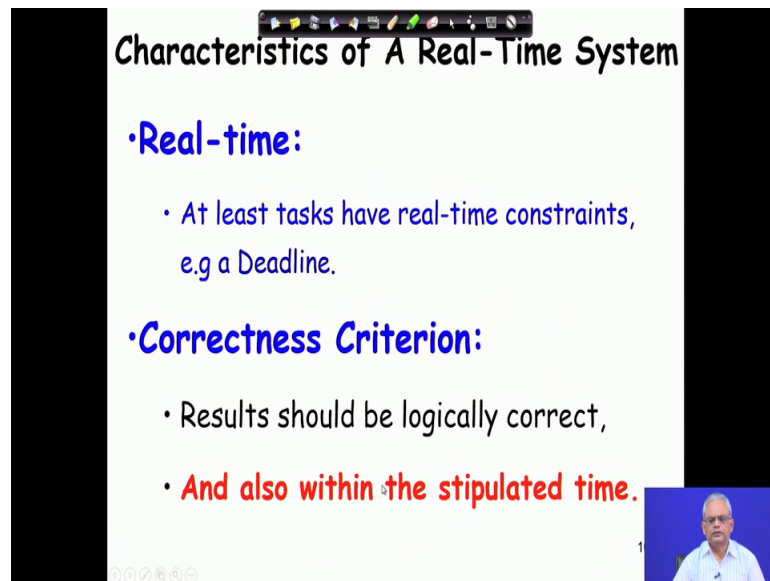
Now, let us ask once you have understood that what is a real time system and we said that one major example of real time systems and embedded systems and then there is a time constraint response that is required for various events. Now, let us ask let us try to answer the second question that how is a real time operating system different from the traditional operating system with which we are familiar to.

So, the first thing is that it somehow has to make the tasks meet their deadlines otherwise the system will fail, if the response is late then the system will fail. On the other hand in a general purpose operating system the operating systems objective is to minimise the response time and ensure fairness for any operating system if you look at just look at its basic objective it is that how fast it can react to commands complete tasks turnaround time and at the same time ensuring fairness, whereas, in a real time operating system we do not really minimise the response time.

As long as the different tasks meet their deadlines the real time system would have achieved its goal, there is no point in completing the tasks very fast that is not the objective. If the deadline is 100 millisecond then it needs to complete by 100 millisecond and there is no reward if it completes in 1 millisecond let us say. So, all the tasks must meet their deadlines and in the process we will see that fairness cannot be a guaranteed.

So, some of the tasks may get delayed, but then overall the task deadline will be met.

(Refer Slide Time: 19:26)



The slide is titled "Characteristics of A Real-Time System" and lists two main characteristics:

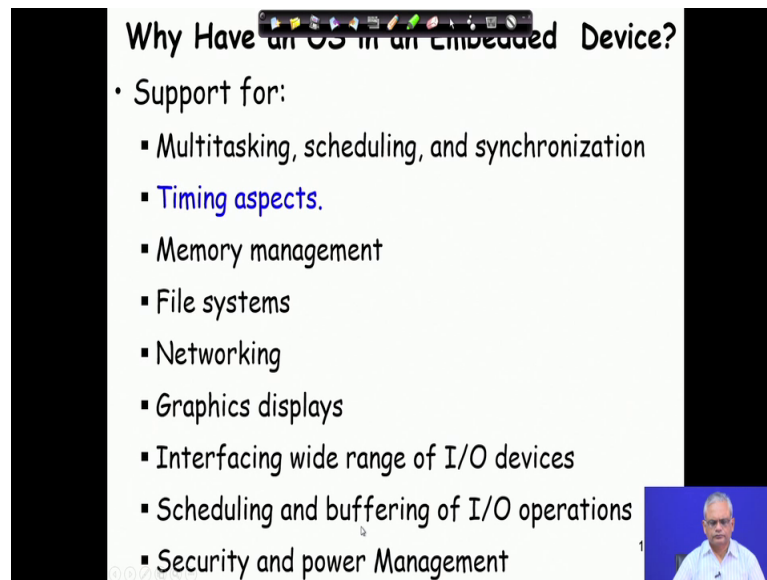
- Real-time:**
 - At least tasks have real-time constraints, e.g a Deadline.
- Correctness Criterion:**
 - Results should be logically correct,
 - **And also within the stipulated time.**

A small video inset in the bottom right corner shows a man speaking. The slide also features a navigation bar at the top and a footer with navigation icons.

Now, let us look at the characteristics of the real time system. So, in a real time system at least one of the tasks would have some real time constraints. Every task may not have real time constraint, but at least some of the tasks need to have real time constraint and also another important aspect of a real time system is that the result not only should be logically correct, but also they should be produced within the stipulated time.

So, just that there are correct result has been produced does not make the system correct, but time is an important criterion and if the correct result is produced after the time required time even then the system will be considered to have been failed.

(Refer Slide Time: 20:25)



Why Have an OS in an Embedded Device?

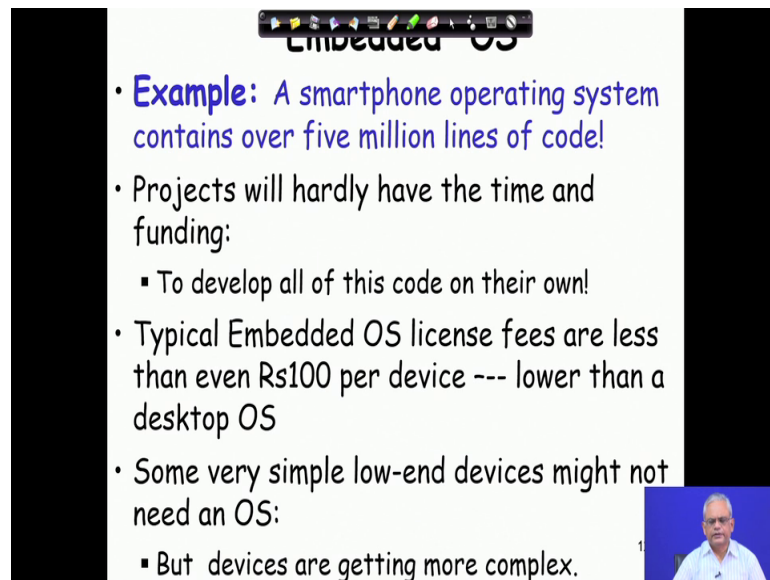
- Support for:
 - Multitasking, scheduling, and synchronization
 - **Timing aspects.**
 - Memory management
 - File systems
 - Networking
 - Graphics displays
 - Interfacing wide range of I/O devices
 - Scheduling and buffering of I/O operations
 - Security and power Management

But then one question that you have might in mind is that embedded systems are really small and they have very little processing power small memory. And then why do we need a operating system cant the programming itself we write so that it monitors and responds without operating system.

For very very simple embedded applications, for example let us say a air conditioner the task is very simple it just needs to monitor the temperature and as soon as the temperature rises it turns on the AC and as the temperature falls to the required level it again switches up the motor. So, that is a very simple task and we do not really need an operating system.

But then if there are multiple tasks synchronization among the tasks you need to manage the memory, like memory demanding tasks, we need to store data in file we need to network to other devices, we need graphics displays we need to interface with a wide range of IO devices, we need to have buffering of the IO applications security power management, etcetera. So, these are easily done using a real time operating system. If your program has to do all these then the program will be enormously large and it will be time consuming and costly to write.

(Refer Slide Time: 22:34)



Embedded OS

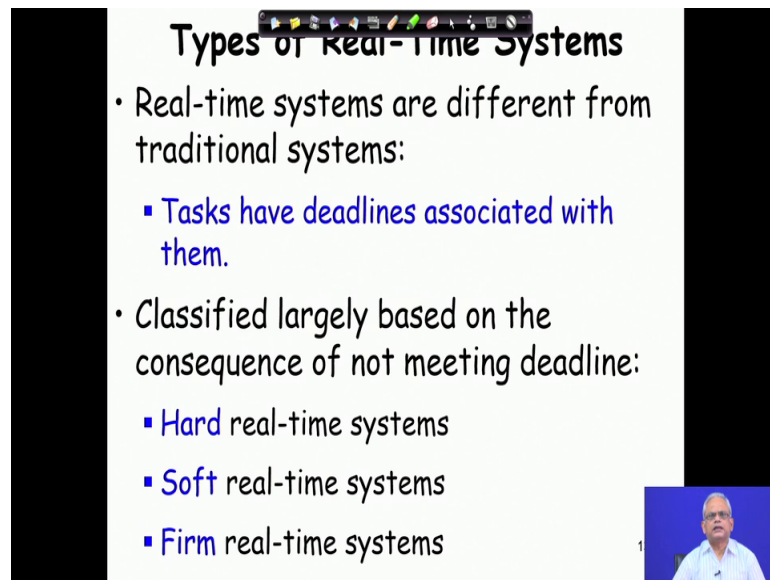
- **Example:** A smartphone operating system contains over five million lines of code!
- Projects will hardly have the time and funding:
 - To develop all of this code on their own!
- Typical Embedded OS license fees are less than even Rs100 per device --- lower than a desktop OS
- Some very simple low-end devices might not need an OS:
 - But devices are getting more complex.

Just to let us give an example where embedded real time operating system is used. A smart phone operating system is extremely large. Several million lines of code and then if the application developer has to write such a large number of codes to do what the operating system does, then the projects will take several years to complete and it will become extremely expensive.

On the other hand if we have an operating system the license fees are very less for embedded devices may be 100 rupees or so, even though the desktop operating systems are expensive in several 1000s of rupees, but for embedded operating system it may be 100 rupees or 50 rupees or even very simple ones may be just few 10s of rupees. So, it makes sense to use a embedded real time operating system rather than really developing the features that these operating systems support.

But as you are saying that very very simple systems we just have a task single task and very simple task without needing much memory etcetera, they might not use operating system I just giving an example of a air conditioner or temperature controller. But then many of the embedded devices are getting more and more complex and sophisticated and all of them they need a embedded real time operating system.

(Refer Slide Time: 24:26)



Types of Real-time Systems

- Real-time systems are different from traditional systems:
 - Tasks have deadlines associated with them.
- Classified largely based on the consequence of not meeting deadline:
 - Hard real-time systems
 - Soft real-time systems
 - Firm real-time systems


Now, let us try to answer basic questions basic question that what are the different types of real time systems and how are these real time systems different from traditional system. That we had already said that in a real time system the tasks are deadlines are associated with them, and even among these real time systems we can distinguish between three, one we call as hard real time system soft real time system and firm real time system.

So, we distinguish between these three based on what is consequence of a task not meeting its deadline. Let us look at these three systems then it will become that what do you mean by the consequence of the task not meeting its deadline.

(Refer Slide Time: 25:27)

Hard Real-Time Systems

- If a deadline is not met:
 - The system is said to have failed.
- The task deadlines are of the order of micro or milliseconds.
- Many hard real-time systems are safety-critical.
- Examples:
 - Industrial control applications
 - On-board computers
 - Robots


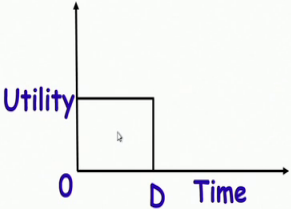


First let us look at a hard real time system. In a hard real time system if a deadline for a task is not met we will say that the system has failed, once an event occurs the required response could not be obtained before the stipulated time then we will say that the system has failed. And the another characteristic of a hard real time systems is that the time restrictions that we give to the task are the range of microseconds or milliseconds, these are not normally in the range of seconds or hours minutes these are micro and milliseconds. And many hard real time systems are safety critical for example, the industrial control applications, on board computers, robots etcetera.

(Refer Slide Time: 26:31)

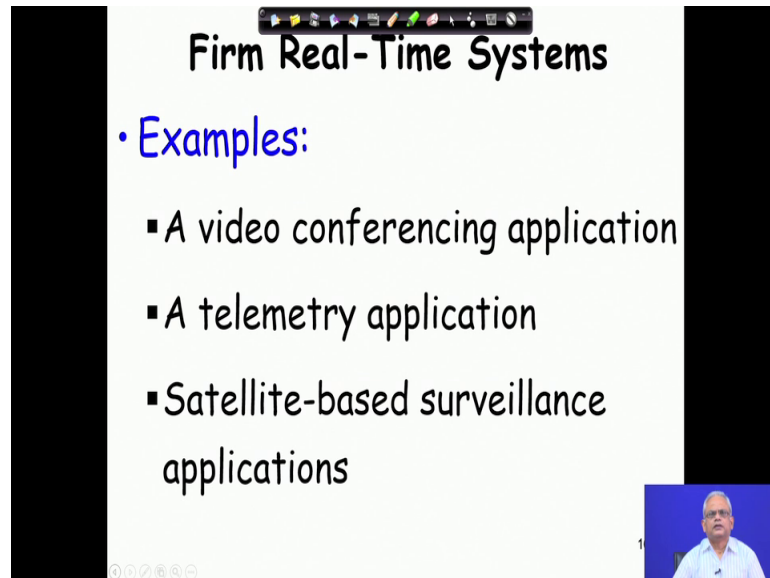
Firm Real-Time Systems

- If a deadline is missed occasionally, the system does not fail:
 - The results produced by a task after the deadline are ignored.



In contrast in a firm real time system if a deadline is not met then the system does not fail only the system ignores the result which is produced late. What is an example of a firm real time system?

(Refer Slide Time: 26:54)



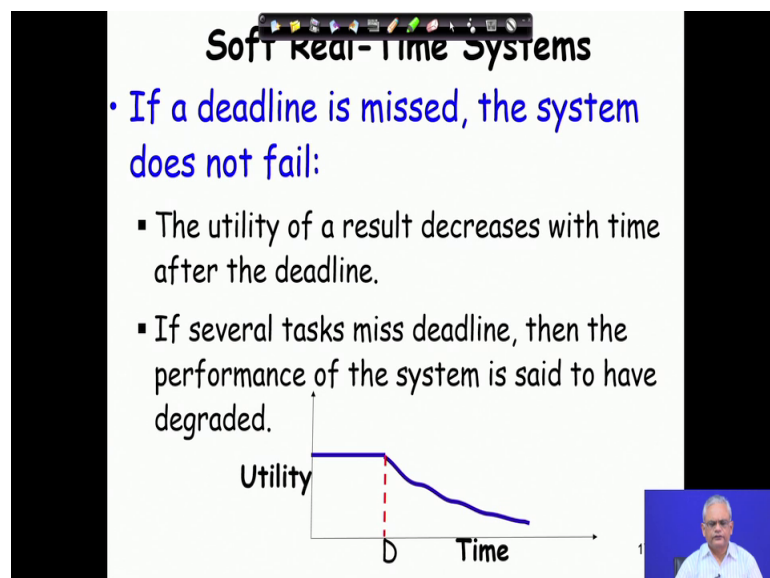
Firm Real-Time Systems

- **Examples:**
 - A video conferencing application
 - A telemetry application
 - Satellite-based surveillance applications

The slide features a title bar at the top with various icons. A small video inset of a man in a blue shirt is visible in the bottom right corner.

A video conferencing application, if a communication data frame arrives late then that is simply ignored, if it stops for getting all the frames before it displays then you will see flicker and it will be unpleasant. So, late frame is just ignored, telemetry applications, satellite based surveillance applications and so on.

(Refer Slide Time: 27:23)



Soft Real-Time Systems

- **If a deadline is missed, the system does not fail:**
 - The utility of a result decreases with time after the deadline.
 - If several tasks miss deadline, then the performance of the system is said to have degraded.

The slide includes a graph with 'Utility' on the vertical axis and 'Time' on the horizontal axis. A horizontal line represents constant utility until a deadline 'D' is reached, indicated by a vertical dashed red line. After 'D', the utility curve slopes downward, showing a gradual decrease. A small video inset of a man in a blue shirt is visible in the bottom right corner.

On the other hand a soft real time system if there is a deadline missed the system does not fail and the only thing is that it just the result becomes less valuable and if a large number of deadlines fail then we say that the system performance has degraded.

So, as this diagram shows that as long as the result is produced within the deadline the utility is 100 percent, but if it s produced after the deadline then the utility gradually reduces. So, what is an example of such a system?

(Refer Slide Time: 28:07)

Soft Real-Time Systems

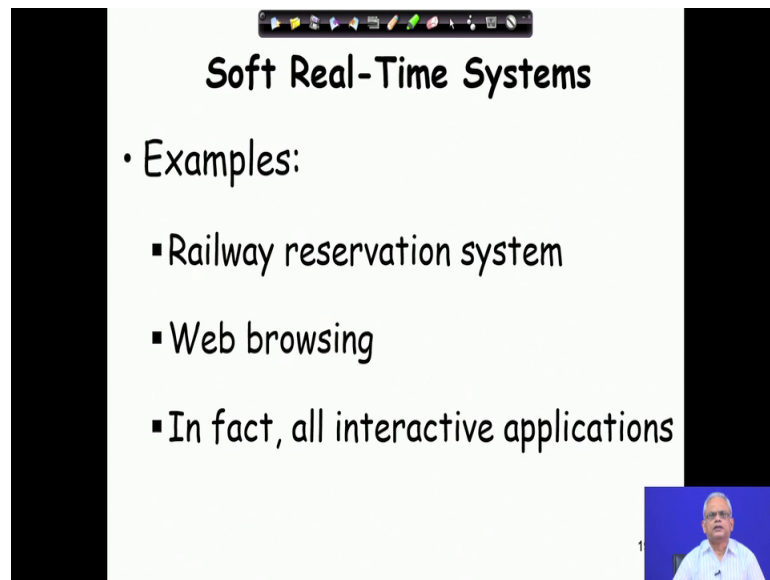
- Use probabilistic requirements on deadline.
- For example, 99% of time deadlines will be met.

Utility

D Time

We will see the example, but here we give probabilistic requirements and deadline. For example, you might say that 99 percent of the time deadline should be met.

(Refer Slide Time: 28:19)



Soft Real-Time Systems

- Examples:
 - Railway reservation system
 - Web browsing
 - In fact, all interactive applications

1

The examples are railway reservation system, web browsing and in fact, all interactive applications where we expect the result to be produced within few seconds 20 seconds, 30 second and then we say that the system is performing well. If it takes minutes several minutes and so on then we say that the performance has degraded.

So, with this discussion about the basic introduction to a real time system we will stop here and continue in the next lecture.

Thank you.