Real-Time Systems

Prof. Dr. Rajib Mall

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

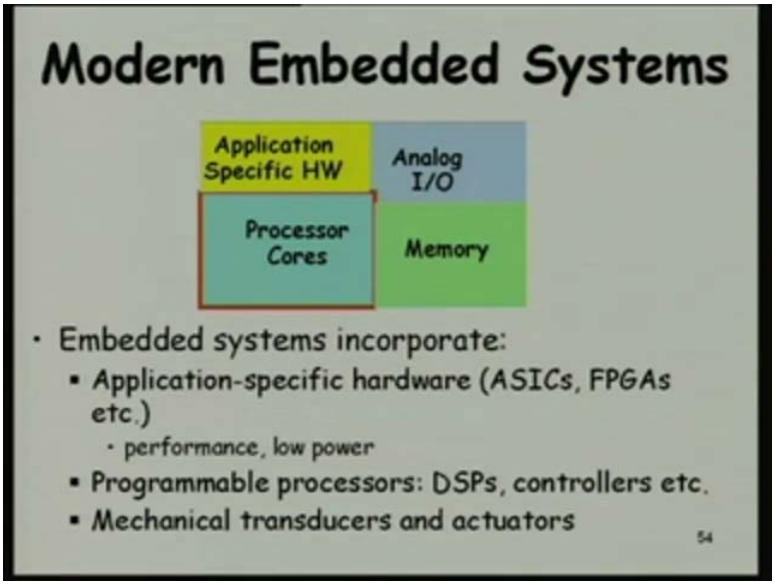
Module No. # 01

Lecture No. # 03

Few Basic Issues

Ok So**,** now**,** let us get started on the next lecture that is lecture 3. We will discuss few basic issues here, which will keep in mind, again an extension of the introductory part.

(Refer Slide Time: 00:40)

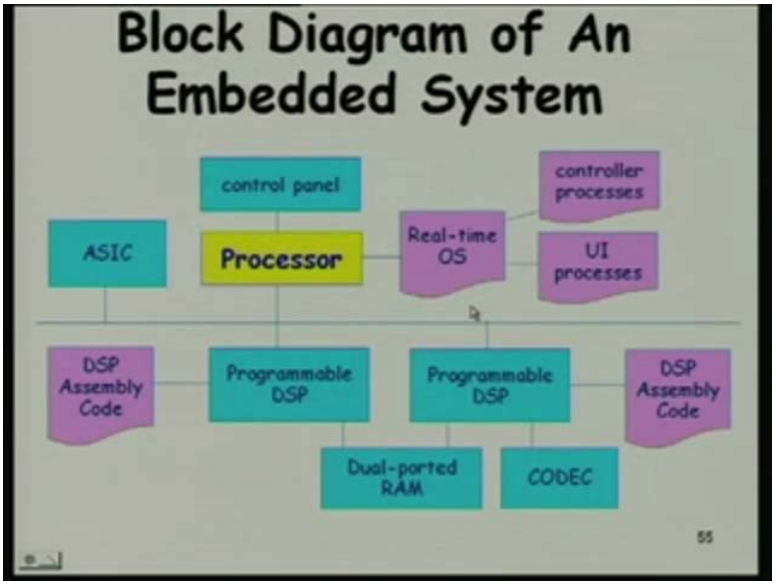


So, let us start, now, just to recollect what we had said that, in a modern embedded system we have processors, need not be one, several processors, and then, we have memory, and then the I O devices - input, output - the sensors, actuators etcetera, these are typically analog. And then, we might have also application specific hardware, designed specific to a application, and this can be ASICs, FPGAs etcetera, let us not bother about this, because not the focus of this course.

The application specific hardware is used here for ensuring performance and consuming low power, and then, we have different types of processors. For example, some might be

signal processors - digital signal processors - some might be controllers etcetera. There might be mechanical transducers, actuators, so we will just keep this in mind, but we will not really discuss this in detail which would be possibly a discussion, in an embedded system course, we will not discuss about this.

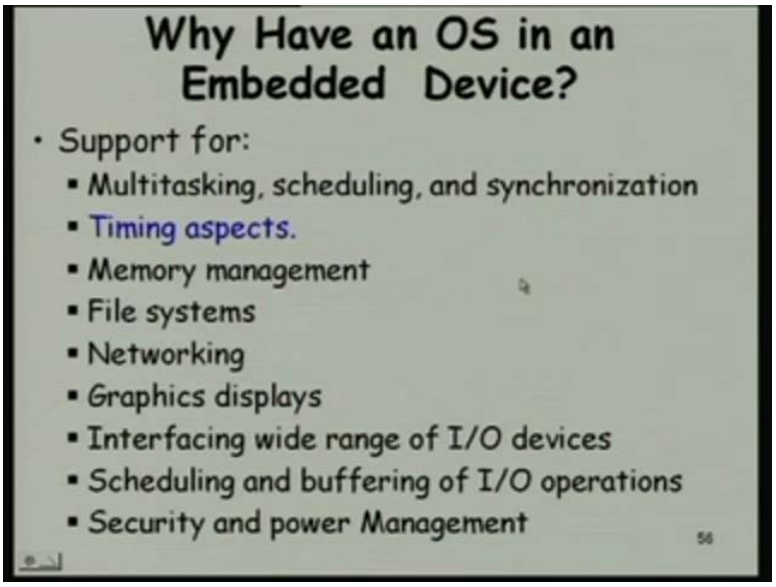
(Refer silde Time: 02:01)



So, this is the kind of block diagram we will keep in mind, that there is a processor and the operating system is running on this, and the user interface processors and the controller processors, these are applications running on the real-time operating system. And then, we have digital signal processors and some specialized processors which possibly have their separate code, and there might be some application specific integrated circuits.

And then, we might have memory specifically for these DSP processors, both of these are accessing this dual ported ram, and we might have a codec for signal conversion and compression, so this is the kind of system that we keep in mind as you proceed.

(Refer Slide Time: 03:09)



But, let us answer one very basic question, before we look into the details of an operating system, said that the system is small embedded system, for example, a small cell phone or a micro oven or a toy or something, now does it make sense to have an operating system there, can't we do a directly program on the hardware.

Why do we need an operating system? We need user inputs, No, user input can be handled by a program written on that. You can always write, you know just like you have done microprocessor programming, you can handle user inputs, microprocessor program, you did not use any operating system isn't.

I am doing an applications program, application.

No, one application will run, what do you mean by new application program.

Sir, The base application O S is the basing because top to hardware all that.

Right.

So, if any new application is coming…

No, but you are developing it for one application isn't? An embedded application is special purpose application.

Sir, if the mobile phone fixes them up, we can add some reprogram there.

Sir, up gradation will be difficult.

Sorry.

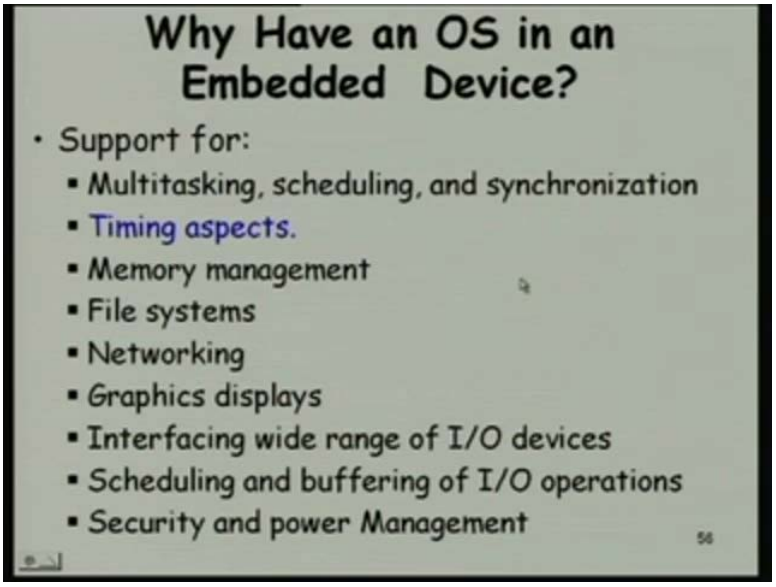
Sir, if the hardware alone is used

Yes.

The up gradation may be difficult.

See, what he says is that, see if you it might be difficult to really make it bug free and all those things, it might be difficult to upgrade. Let us see, there are many reasons, why we need an operating system, say, let us look at them one by one.

(Refer Slide Time: 04:50)



See, one is we said that this systems have multiple tasks, so we need to support multitasking and we need to have some scheduler for this, and then we have issue some resource sharing and synchronization. So, if you want to implement all these as application program, you want to rewrite that; you are unnecessarily incurring additional cost which you could have just got these features by using a standard operating system with very little cost.

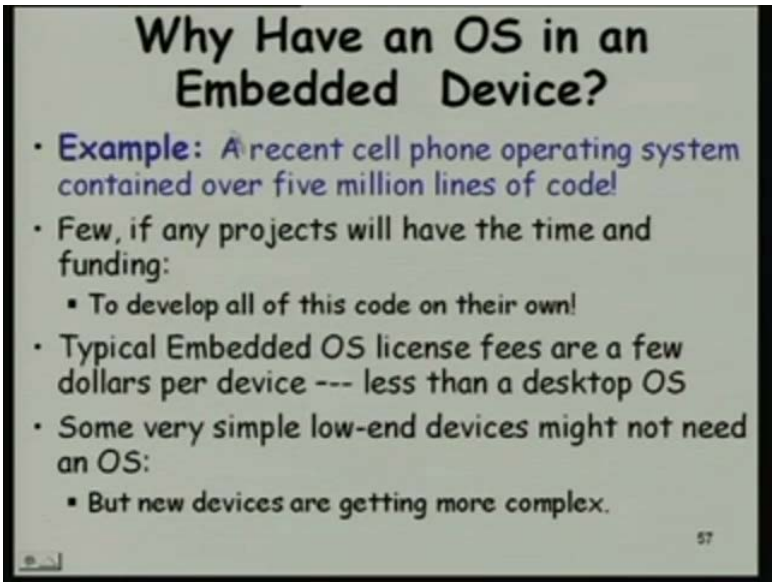
Similarly, the timing aspects, how do we support timing aspects, if you are writing your own program without an operating system, how do you ensure that it will run. And memory management, because these are basic functions in operating system and any application you write, there, it has to do memory management isn't? Because, data it has to read, store. And what about file systems, how do you keep track of permanent data and read them, need a file system. Networking, because as you are saying that all these devices are now been networked. How do you support network, would you like write your code for and the network protocols and handling the packet transmission. What about graphics displays, how do you do that without an operating system.

Suppose, you want to interface with different I O devices, will you write your own I O driver that will be very expensive. Then, buffering and scheduling, the I O operations, what about providing security and doing power management, so all these are features of any operating system, real-time operating system. And if you are thinking of writing an application without using an operating system, it will become extremely large, development time will be too much, cost will be too much, and the product will not run.

So, for every non trivial embedded device we need an operating system, is this point clear. There are many reasons why need an operating system, because these are basically multitasking and we need to have proper schedulers for this, resource sharing between tasks is required, as you will see that some results are produced by something, used later by another task, or maybe they just work together to produce some results.

Memory management, file system, networking, graphics displays supporting IO devices, security, power management, all these are provided by operating system. If you are thinking of not using an operating system, you will be unnecessarily incurring additional cost, time.

(Refer Slide Time: 07:58)



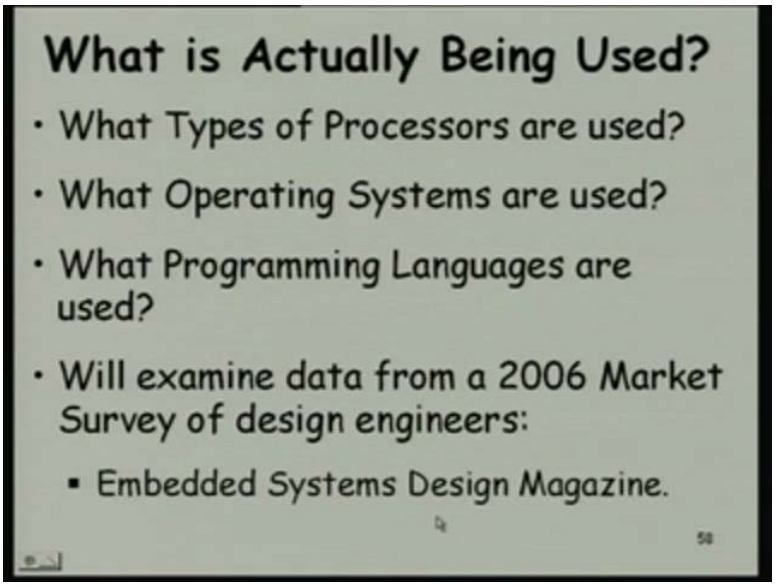
Right, let us proceed further, just consider a cell phone operating system, this is one topic we will discuss later, a kind of operating systems that are been used in cell phone, commercial phones. The code is over a five million lines of code that is the operating system that is being used. Now, suppose a cell phone company says that, see, we want to reduce cost, let us assume hypothetically, because when you buy a commercial operating system and per product you will have to pay license fee. Now, the license fee may be typically small, may be just few 10 of rupees or may be just fraction of a dollar per device, per cell phone.

Now, if you do not use the operating system and say that, see, our programmers will write the basic features necessary to support that. Now, just see how much time they will take to write that five million lines of code to make that cell phone operate satisfactorily, and also the fees are not very high, per device, the license fee that is paid is just few 10 of rupees, and also there are free operating systems.

But, what we need to keep in mind here is that this kind of operating system they do occupy some space, they have overhead. So, if you are thinking of extremely simple devices like, let us say, a air conditioner where it just samples the temperature and either switches on or off - samples temperature switches on and off - that simple activity, then we do not need a operating system there, the event is just one temperature being sensed and then switching it on and off, that simple program can do. But the kind of devices that

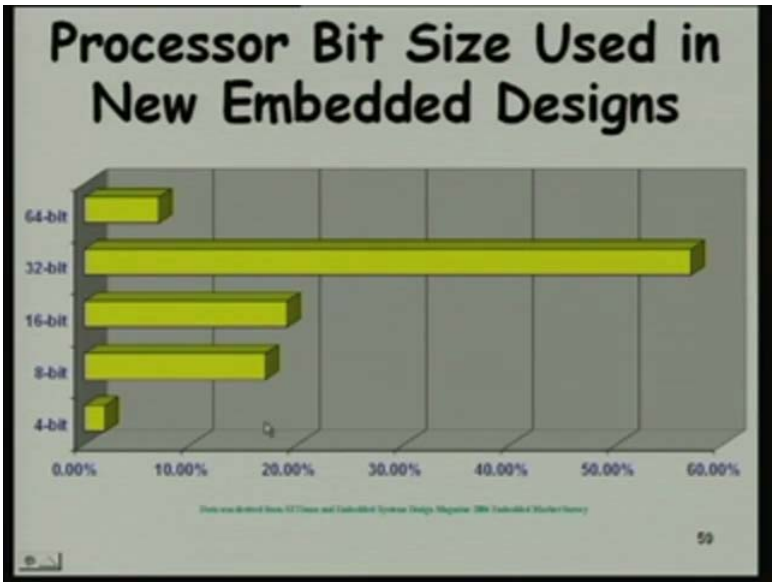
are appearing now becoming more and more complex, and most of these devices we will use operating systems.

(Refer Slide Time: 10:15)



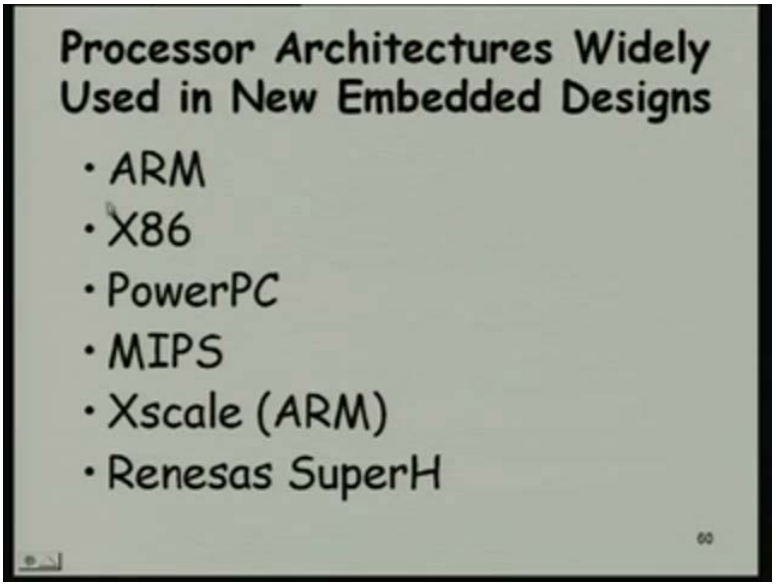
But, before we actually start our discussion on the operating systems etcetera, let us look at these basic issues that what kind of processors are used, what kind of operating systems are used, what kind of programming languages are used. We will examine some market survey that appeared in the embedded system design magazine; now, let us look at the results.

(Refer Slide Time: 10:46)



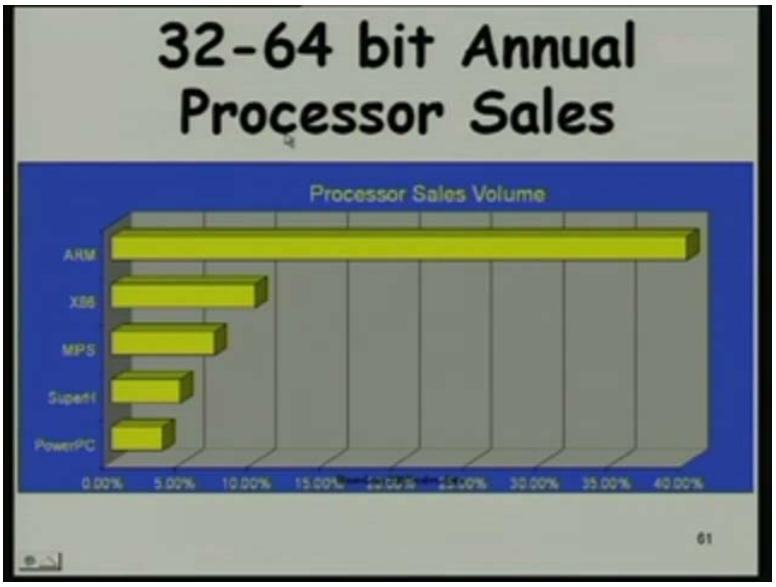
See, the processors that are being used in these embedded designs, vast majority is 32 bit, but for lower end application we still have 8 bit, 4 bit and 16 bit applications, and for some applications even 64 bit processors are being used. But, may be in future as the embedded systems become more and more complex, maybe we need more powerful processors and powerful resolution, right data resolution, data size might need a 64 bit processor.

(Refer Slide Time: 11:31)



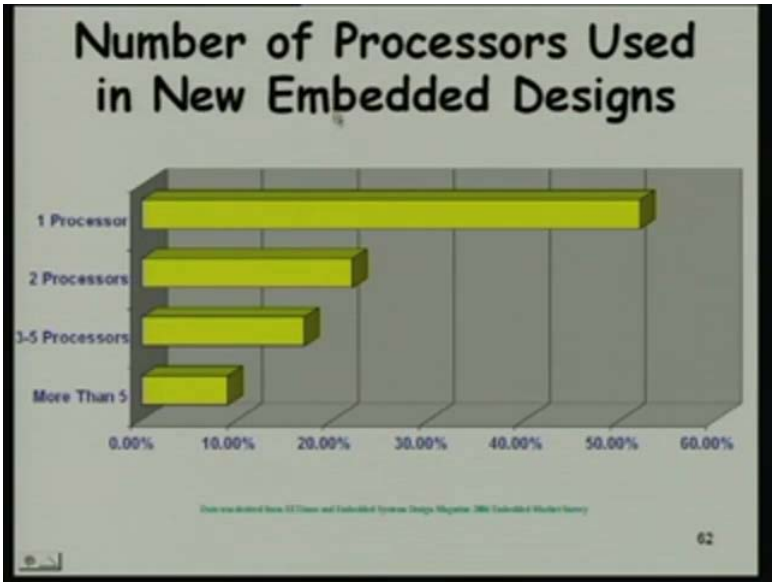
The kind of processors that we are talking of, in this kind of applications, a large chunk of the processors based on the ARM processor, PowerPC, MIPS, Xscale which is again a type of ARM processor, but we will see that a vast majority in cell phone and all those are ARM processors, which is again based on the MIPS design. I hope in your architecture course, you had the architecture course isn't it? So, in the architecture course, the MIPS, the design of this is or the arrangement of this is that they do not really manufacture MIPS processors, they just license it to different vendors, for example, Sony corporation might be using MIPS processors, Toshiba might be using, let us proceed.

(Refer Slide Time: 12:42)



Now, let us see as far as the sales volume is concerned, the arm processor has the maximum market share as you are saying that many of this mobile phones usually based on arm processors.

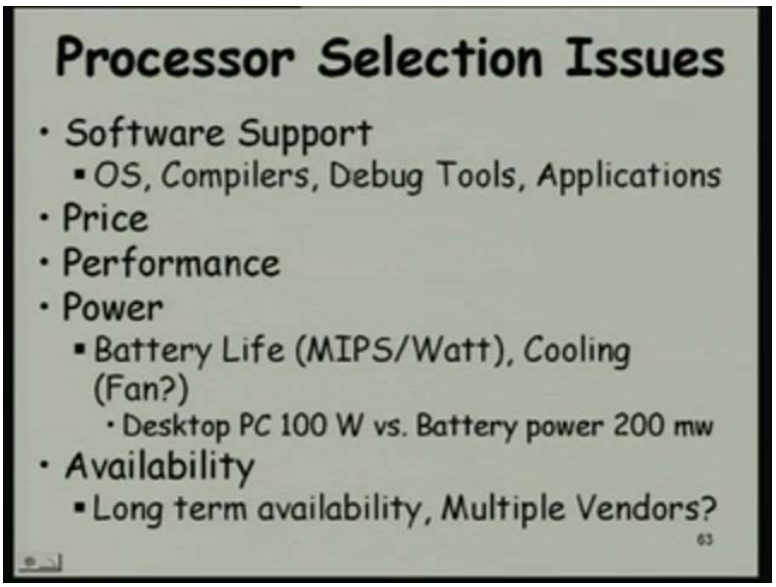
(Refer Slide Time: 13:01)



But what about the number of processors used, we have majority one processor system, but we do have two processor, three to five processors and more than five processor system.

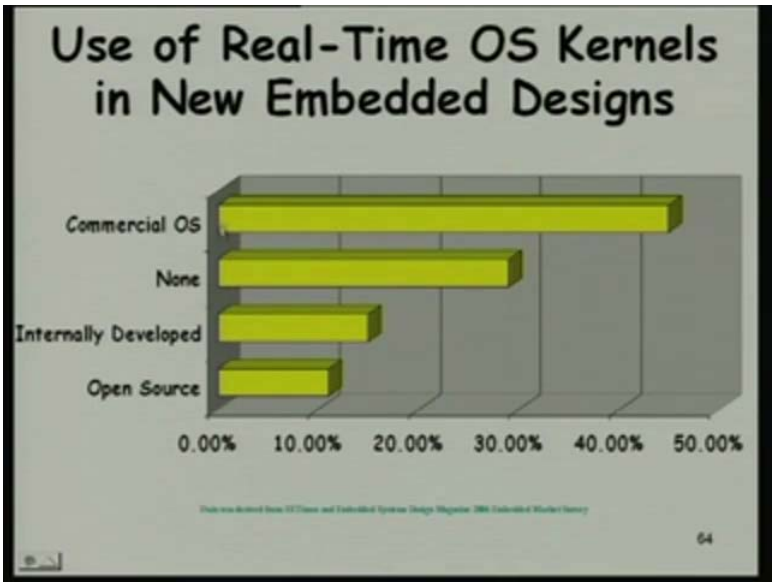
So, in our discussion, later we will keep in mind about multiple processors been used in an application. And that trend is possibly be - I mean - more and more number of processors, more and more number of threads for processor, that is the trend.

(Refer Slide Time: 13:38)



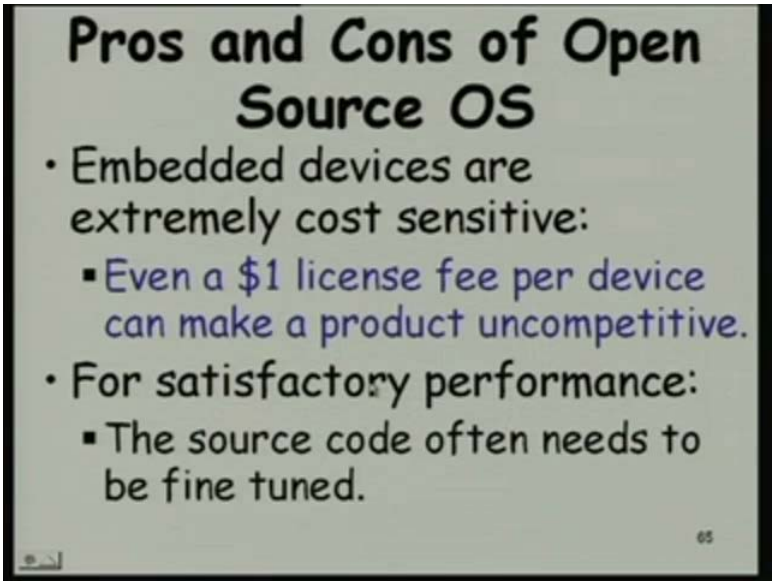
Now, let us look another basic issue that, if you are developing an application how do you select the processor, kind of operating system it supports, because all operating systems do not run on all processors, kind of compilers, debugging tool, applications you need, the price, performance, power it consumes, because for this kind of applications the battery life is important, millions of instructions processed per watt, is there a cooling facility available, whether you can get in sufficient number and time, those are issues, not spent time on this.

(Refer Slide Time: 14:24)



But, what about operating systems that are being used, see, a large majority more than 50 percent, they have used commercial operating systems. And the simpler ones about 30 percent, slightly less than 30 percent, they use no operating system, because very simple systems. Internally developed operating system for simpler applications and then, open source operating system. This come without cost right, but this is a growing segment actually.

(Refer Slide Time: 15:11)

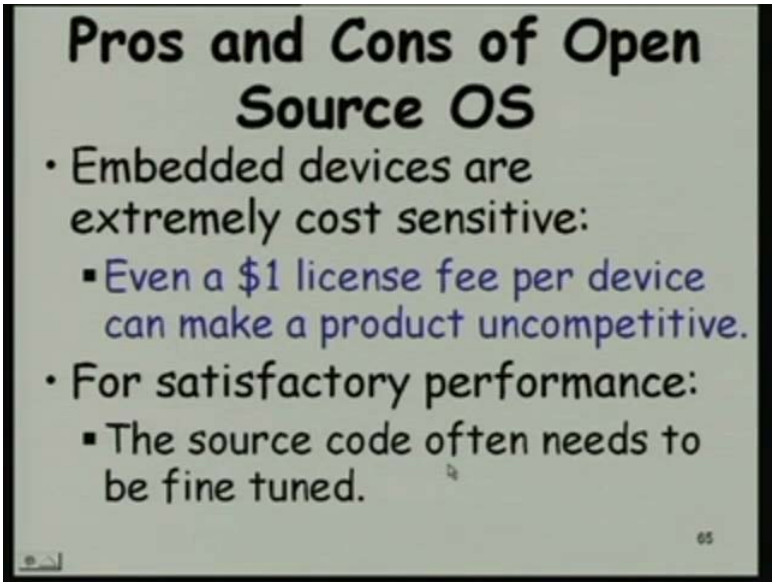


But, let us see, what are the problems with open source operating system because, as we proceed we will also discuss different types of open source operating systems. So, let us look at one basic idea which will remember throughout the course, see, one thing is that, that goes in favor of open source operating system is that devices are extremely cost sensitive, because the total cost is let us say 500 rupees and out of that if you pay some 20 rupees for operating system license, then another competitor might actually under price you by not using an operating by using a open source operating system.

See, for a commercial operating system paying a license fee, you will just try to use an open source and reduce the cost, very competitive market. So, that is one thing going in favor of operate open source operating systems, and people actually exploring to use open source operating system.

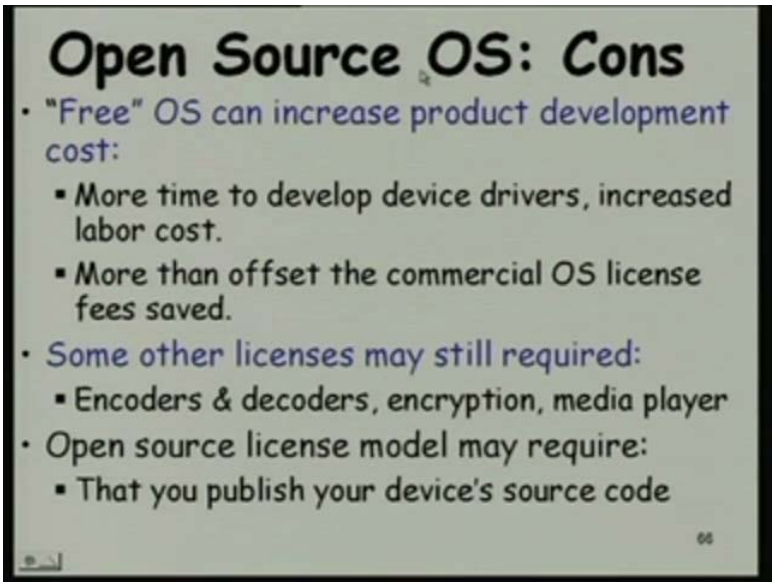
Now.

(Refer Slide Time: 16:21)



But for satisfactory performance, this open source operating system need to be fine tuned. You can just to get it and use it, right you have the code, source code is available, these are open and you need to fine tune, but that is not a problem, could have also used without fine tuning.

(Refer Slide Time: 15:58)



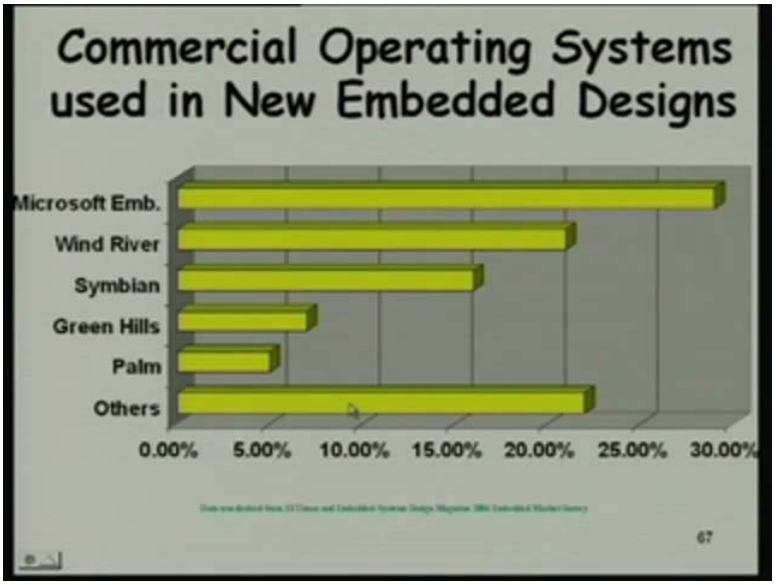
But, let us see the problems, say, cost is one thing, you could reduce cost, but let us see the problems, why there are majority still going with commercial licensed operating systems. See, one thing is that even though you got the operating system free, but these free operating systems - the open source operating systems - many of the new devices they do not get drivers - device drivers. So, possibly you will have to write device drivers, which will increase the labor cost. Whereas, the licensed one, as soon as a new hardware becomes available - I O device, a new device becomes available - they immediately develop the device driver and support it.

And there are other problems also, one problem just see here, that as part of the open source license, it is not that you will just get it, change it and use it, you will have to sign a license which says that whatever you develop based on this operating system, you will also make it in the public domain, you will make it open.

So, very few developers - product developers - will be agreeing to put their code on the open platform, because of business reasons, and for other plug-in software still needs licenses, they are not part of this open source, for example, in media player encryption encoders, decoders. So, possibly these two, that you do not have many device drivers support, a recent device driver support and also you would have to abide by the open source license, these may be deterrents for using a open source in a commercial product.

But, if you are using the operating system as it is without much difficulty - I mean - without much tailoring or rewriting then, it make sense, but again to get programmers to understand that code and write it and support to have a problem there in the operating system, how do you who will support it, so those are the issues, let us proceed further.

(Refer Slide Time: 20:02)



So, out of the commercial operating systems, let us see the market statistics, actually slightly older statistics 2006 published in literature. The Microsoft has a large share here, Microsoft embedded, and then we have wind river system, they have several operating systems under the wind river which are also very popular, and then we have Symbian, Green Hills, Palm OS, these are some of the names we used as we proceed in this course, and of course, there are the free operating system, and these are the small players in the commercial operating system area.

We will look at it in more this thing detail, the typical products that are there, the popular products not typical, the popular products under from these commercial organizations and their features, how they compare for using in a typical application, we will look at those features later.

Now, what about the programming language? See here, large majority is with C language, it is not really C, but a subset of C, because they do not use pointers, dynamic memory allocation, this features are not used in a typical embedded design, because their

sources of error. So, the standards here is, they have defined a new standard for embedded system programming, where they define the subset of C that is being used, C plus plus is also quite popular.

Sir, what is that subset of C called here?

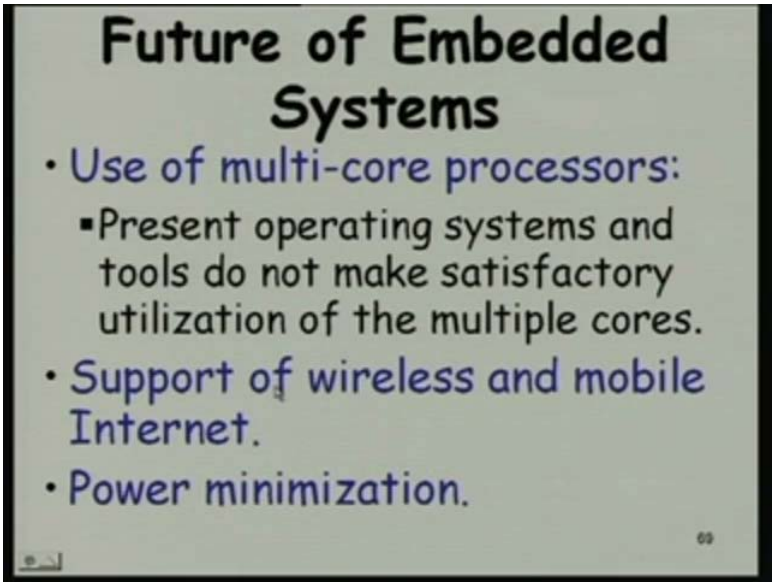
I think, it is called a standard, I will just give you the exact organization which produce the standard for embedded programming, it is a the syntax and the schematics, subset of the C has been specified, which are being used by organizations across, I will give you the organization and the standards name.

Is it misra.

Misra standard yes, but misra is it, that for I think automobile.

Automobile, but what about the general embedded? We will just discuss that later, I do not recollect now, the name of the standard and who has made the standard. So, C sharp is also been used, java of course, because of its different technologies that are available with java, especially in the mobile environment right - mobile phone environment. Assembly languages are also used significantly and other programming languages.

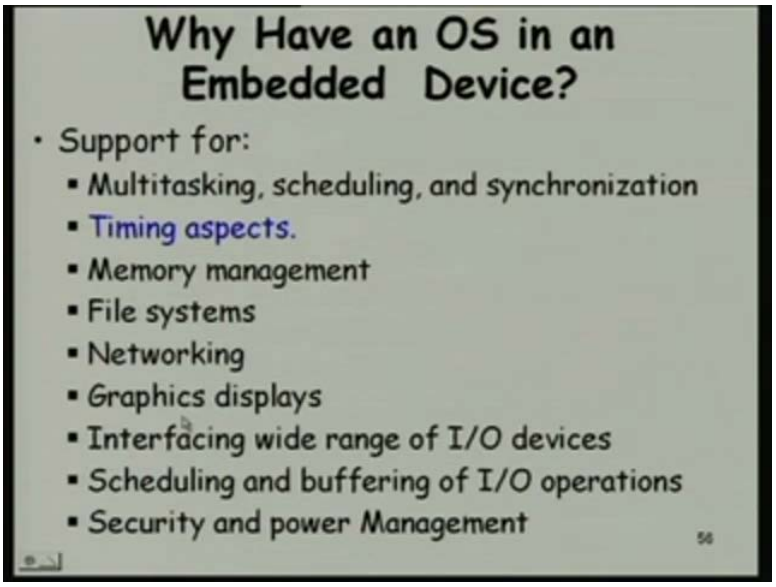
(Refer Slide Time: 22:22)



So, as we proceed, we will hear less about embedded systems, this is possibly one of the final slides that we are hearing about embedded systems. So, in the future what are the trends, because this we will also have to keep in mind, as we discuss about the software issues, more and more they are using multi core processors, where in the same semiconductor packaging, same processor will have multiple cores and. The challenge that it possess for, our discussion in the subsequent lectures is that, the operating systems and tools, how do they make effectively they use the multiple cores available.

And support for wireless and mobile internet, so how would one support internet while on move, power minimization, so these are some of the issues that will become more and more prominent in the coming years.

(Refer Slide Time: 24:50)



Now, let us start, possibly we will not discuss about embedded systems and the basic issues. Let us look at the types of real-time systems that will be discussing, one thing that we have so far said, is that every task here in a real-time system had some - I mean - the real-time tasks, I am not saying every task in the system, but the real-time tasks are those which have deadlines or some form of timing constants associated with them.

So, based on this we can classify the tasks as hard real-time, soft real-time or firm real-time, very basic issue. And later, if you say that I have done a real-time systems course, somebody might, the first question asks you is that what is a hard real-time system, how

is it different from a firm real-time system, or how is it different from a soft real-time system, give examples of this right, these are basic questions, in any if after having done this course anybody can ask you this question, so let us look at it.

In a hard real-time task, even if a deadline is not met, just one deadline is not met; we will say that the system is failed. And another thing that we need to remember is that the deadlines that we are talking of are hard real-time tasks, are of the order of micro to milliseconds, never in seconds or minutes, all these hard real-time systems that the examples we had seen several of them, starting from Robo's to a chemical plant to on-board computer, all these. The deadlines are of the order of a few microns or maybe few milliseconds not more than that, and many of this hard real-time systems are safety critical.

You had seen several examples, but these two are important points that are the basis on which you will distinguish that if one failure has occurred, what is our opinion of the system? Is it failed or can it still continue to work, and then what is the kind of the deadline we are talking of. A firm real-time system is one where even if sometimes a deadline is missed occasionally, the system does not fail, but only thing that will happen is that the result will be rejected, the result produced after the deadline will be rejected.

So, we can draw this kind of diagram and say that, see, as long as the result is produced within time d, we have the result will be used, utility is 100 percent - I mean - the result will be useful to us or to the system, but after d the result is produced, the utility of the result is zero will be thrown right, will be discarded, but the system will not fail. But if it fails once in a while it is all right, but if it just every task fails then, we will have to say that it is failed.

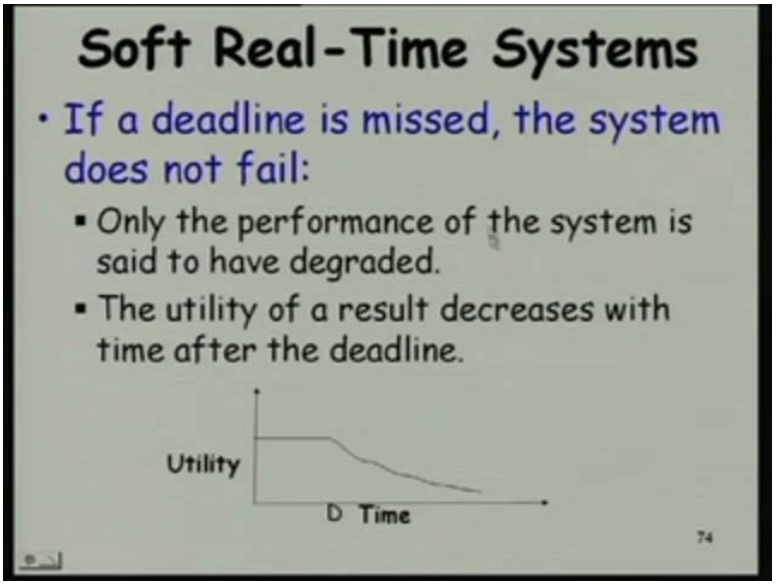
What about examples of a firm real-time system, typically, all video conferencing and multimedia type of applications are firm real-time system, suppose, you are in a video conferencing mode, and then a frame is missed or frame could not be processed, you would hardly notice it, because there are 50 or 60 frames being repeated every minute, I mean, the number of frames repeated depends on the quality of your video conferencing system, quality is poor might be some 20 or 30 frames being repeated.

High quality video conferencing, you will have many more frames that are repeated per minute. So, one or two frames getting missed occasionally, the user will hardly notice that, and as soon as a frame arrives late or some for some reason could not be processed in time, you do not show that later, if you show that later then, you will see a glitch or you will see a problem in the picture, it just discarded, a frame which could not be processed in time, will be discarded, and the users will hardly notice that.

So, the result which could not be produced within the deadline, before the next frame could be processed, or if the repetition rate is something, it could not be processed within that repetition rate, then it simply discarded, not displayed right, so that is an example. You can have telemetry applications, if a new data becomes available do not need the old data, it might discard it, before the old data comes the new data which has come and processed will discard the new data without much difficulty, because after just measuring, something for weather or something.

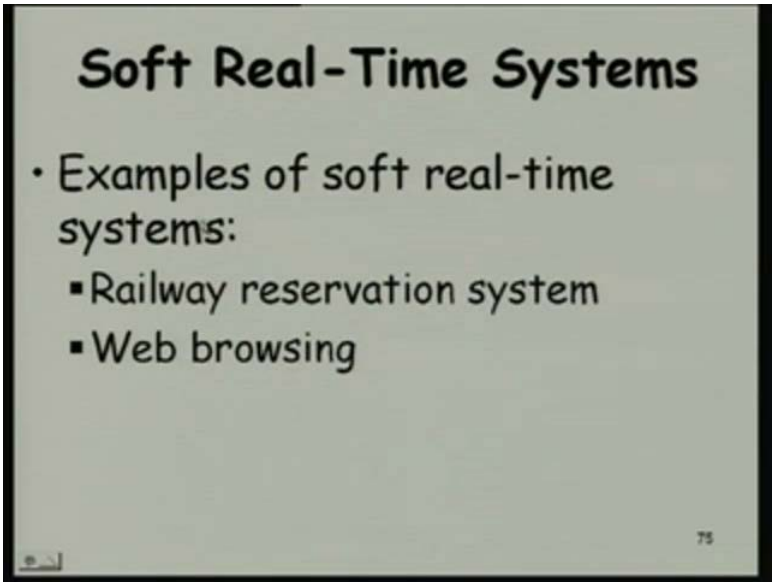
Then, satellite based surveillance application, we are doing surveillance, you know that getting signals and processing and seeing whether there is a movement of army or what is the crop that kind of surveillance. So, if a data could not be transmitted in time, we have the new data for surveillance application, we will use the new data, discard the old data, so these are some of the examples of a firm real-time application.

(Refer Slid Time: 29:56)



But, what about soft real-time, a soft real-time application, if a deadline is missed, the system does not fail just like a firm real-time system, but the difference is that the result still is used even after a deadline has passed, the result is not thrown still as diminishing utility value. So, the utility of the result decreases with the time after the deadline, is that ok, the difference between a hard real-time, a firm real-time and a soft real-time, but what are the examples of soft real-time.

(Refer Slide Time 30:45)



Let us look at a railway reservation system, let us say, you are standing in the counter for a railway reservation system and the clerk there, took your application for some train journey and just keyed in the detail, and if the system does not respond, you just keeping on waiting waiting. See, if it comes, let us say, within 30 seconds or 20 second, say, that the system is very fast and you come with the ticket very happy.

Now, let us say, it delayed 3 minute, yes, given the data still you have to wait 3 minutes and then finally it printed. So, we will say fine, it was slightly response time is not good, now what if keyed in the data and even after half an hour still nothing has come out, what will you do, you will just walk out, lose patience that possibly it is not working and you will come out.

So, as long as it is in the human response time, see, if it is within 20 second or something we will not notice 20 second, say, that it has come instantly, and you will come happy

saying that response is very fast, as soon as I gave the request immediately ticket is printed and delivered.

Similarly, in the case of web browsing, you just clicked on a link and then, if it what appeared to you instantly, see, what appeared to instantly is not really instantly, it is not in microsecond or millisecond, several seconds. If it comes in several seconds, you say that see is good; it is come very fast, but let us say you waited have to wait for 3 or 4 minutes, so that system is slow or that site has some problem.

Now, after 10 minutes it does not come, so you say that see there is problem, it is not usable. So, here just see that, if it has come within some 20 second or 30 second, say that the system is working very good, and then you are saying that the response time has decreased, the system is not as responsive, but you are not saying that the system has failed.

So, in that sense, every application that you will use is actually soft real-time - I mean - every application in your desktop or in many of the everyday life you use are soft real-time. For example, you are doing a word processing, and sometimes you make a change on the word processor immediately reflected, and sometime you just made a change and its taking time, possibly it is doing something internally, saving some file or doing some logging or some activity, takes little bit of time. You do not say that the system has crashed or some, you wait right and if you have to wait very frequently and for longer duration, say, that the response time of the system for some reason is not good.

So, most of the task that we are familiar are soft real-time, the firm real-time and the hard real-time are the one that we are going to discus in this course. Soft real-time occasionally we will talk of, that in addition to hard real-time and firm real-time, some soft real-time tasks also systems might have to support - real-time systems - but majority of the tasks that will talk of are either hard real-time or firm real-time.

Now, is there any task which will be non-real-time, what do you think? See, we talked of a hard real-time, a soft real-time, firm real-time, but can a task be non real-time? Yes, anybody would like to answer, can a task be non real-time?

Like a complex inversion and then, calculations we have not concerned about the time we just concerned about the result.

Ok,

He says about a simulation application, it runs days together, but sometime it might, so you do not know that, say, maybe you will come tomorrow and check, not running come next day and so on, any other application you have in mind non real-time application, non real-time tasks let us say,

(( ))

Ok.

(( )) if it is only if it is two days there we say it is, ok.

Or let us say in the salary section, they give the printout for salary they gave, the printout takes time, went for a tea and came back, so if it has completed fine, not completed it is going on they will check again, so time is not a matter there as long it is going on printing.

So, one thing that distinguishes between the soft real-time and the non real-time, can you identify that? See, soft real-time also, time is of very little concern isn't it. So, what is that distinguishes? See, here you went for a coffee break and came and found whether it is printing or not printing, and then you again checked later whether it has completed.

But in a soft real-time also you just waited with patience, checked whether it is printing the ticket or no right, so what exactly is the difference between a soft real-time and a non real-time task.

In a soft real-time ,we had been some in a, non real-time system we can you can do some other task by or task has been completed, but in a soft real-time system we are struck into that task while the task (( )).

In other words, what he says is that, in a soft real-time, the time scale that you are talking of is couple of minutes whereas, in a non real-time task, we are talking of time in hours.

So, the dimension of the time is possibly the difference, because if the pay role is suppose to have been printed within an hour, and is still going on, so you say that it has failed, its printing possibly something junk, you go and check right. So, the scale of the time is actually the difference between a soft real-time and a non real-time task.

Sir,

Yes,

Sir, can we talk about the utility of the output that we get, like we are saying that in a soft real-time system, the utility suppose to decrease.

Yes.

Of the output that we get.

Yes

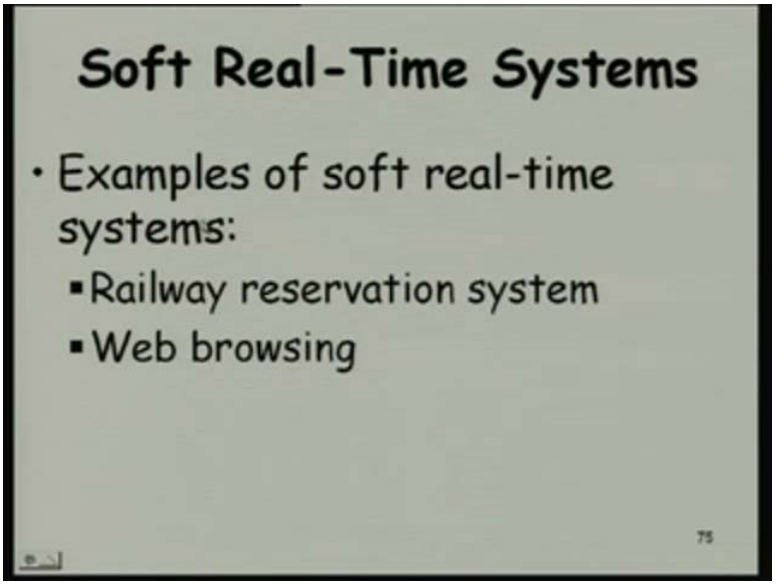
But for non real-time task like, if the printout comes after two hour (( )) utility of the printout of the output it is not (( ))

No, see, it is taken your time, utility of the result has diminished, just like in a ticket counter - railway reservation counter - you got your ticket, after let us say 2 minute, so the utility is less, it is not you know you have to wait, you are not happy right, so the utility of the result is less. If it came instantly, you went came back happy saying that it is a responding, otherwise you are saying it is in a degraded performance. Same thing here, that if it appeared in let us say 15 minute printed, and we are expecting an hour, say that it has printed very fast, and after 1 hour if you come and check it is still printing, you will again say that somehow it is very slow, let me come after some time, so degraded performance same thing, but the scale of the time one that actually distinguishes these two.

Or maybe let us say a email, in a good time it just reaches instantly, may be in few seconds, but let us say, it can also take 5 minutes or it can take half an hour, but if it does not take reach by a day or so, you say that see it is not going to reach. So, the time scale that you are prepared to wait for the task, that is, actually the one we can say

distinguishes between a soft real-time and non real-time and say that email is not a soft real-time task, it is need not complete in a minute or couple of minutes, it can take several hours right. So, this is the scale of the time is the one which really distinguishes between these two right, let us proceed further.

(Refer Slide Time: 42:27)

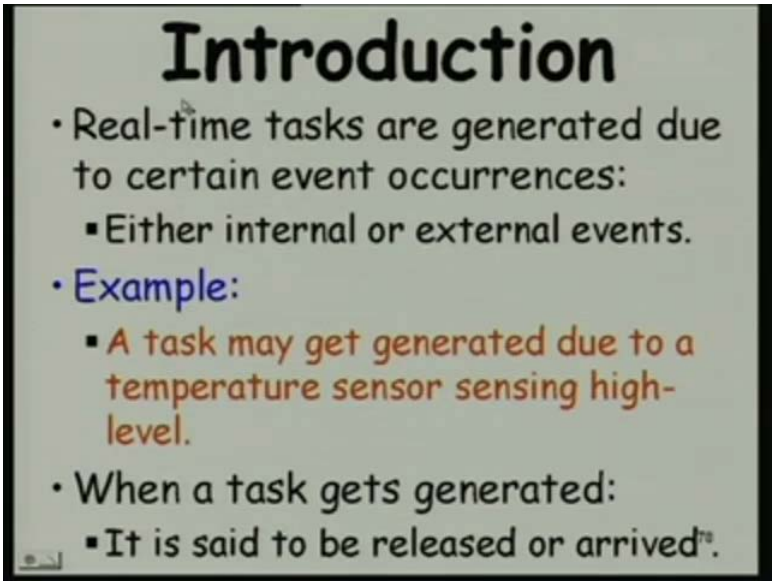


So, any interactive application we can say is a soft real-time task, see, printing is actually a not a interactive task, we just pressed a switch and then, we are waiting for the result similarly email, but if you are interacting with the computer, entering data looking at the response, entering again, so then those are usually soft real-time. Now, we can also say that a task in addition to being a hard firm or soft real-time and also say that whether they are periodic sporadic or aperiodic, is the another dimension of classifying tasks, we will use this terms very frequently as we proceed.

The periodic task is one which repeats after some fixed time interval, keeps on occurring based on some time, just look at the term time. So, somehow, some clock is there giving a time interrupt and the task is raising. For example, you are trying to sample temperature every 100 millisecond, so there will be a clock, which is giving a interrupt after 100 millisecond, and based on the interrupt, the task of sampling, the time is arising, is that ok. There is a concept, there has to be a clock in every periodic task which as soon as it gives interrupt at periodic times, then the task is getting fired or it is arising ok.

Now, a sporadic task on the other hand, occurs at random instants, for example, a fire condition or let us say temperature becoming very high, you know suppose, we just monitoring the temperature, the sensor will monitor temperature, and it will send a signal as soon as its more than some, instead of sampling it periodically sends a signal once it becomes exceed some value, or let us say a fire condition or a smoke condition being sensed, so these are sporadic tasks. You cannot say when they will occur, at random instants they may occur.

(Refer Slide Time: 42:32)



Now, there is another task, another type of task, we will also use this term aperiodic. So, these are also sporadic tasks, in the sense that they occur at random instants, but the difference between a sporadic and aperiodic, as we leaves is that. A aperiodic task we can have several of them arising even at the same time, for example, different users are there in their you know giving inputs to the system, so they might give input very random time or they might just give at the same time, all input has come.

Or let us say we are monitoring fire condition at different places, and let us say, fire can occur at different places at the same time, and different signals will come that may be a sporadic, so, sorry, aperiodic. So, a sporadic and aperiodic are similar terms with a small difference, one there is a minimum difference from one event to another event, whereas in a aperiodic the events can occur any time and even at the same time, but just to summarize this classification on real-time task, a periodic task is driven by clock, the

clock event is the one which is important for a periodic task, whereas, for sporadic and aperiodic the events are not driven by any clock, can occur any time.

Now, let us look at some basic issues in task scheduling, because we said that this is one issue which will be there in every operating system that we talk of, and this is the primary means by which the deadline will be ensured.

So, do you have a class after this? Ok

So, I think, we will just take up these topics in the next class about task scheduling, so we will just stop here, ok thank you.