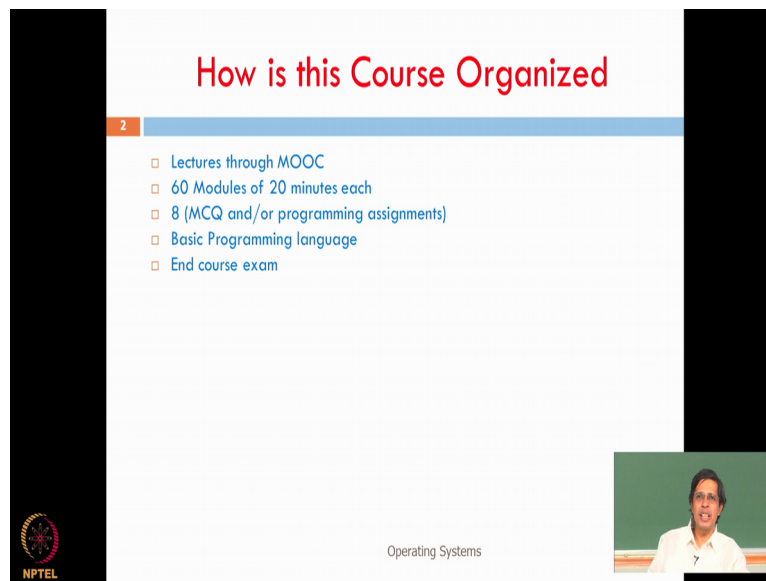


**Information Security-3**  
**Prof. V Kamakoti**  
**Department of Computer science and Engineering**  
**Indian Institute of Technology Madras**  
**Basics of Unix and Network Administration**  
**Operating System Introduction**  
**Mod01 Lecture 01**  
**Module 1: OS-Definition, Role and Types**

So welcome to information security-3 course on basics of UNIX and network administration. This is the third in the series of information security course has offered through this mooc platform from IIT Madras. In the first course we just had an full introduction to information security from a systems perspective. What it means to build a secure system? What are the different definitions etcetera. There were the thing that was covered information security 1, which was 2 years before. Last year (0:42) very we had as (0:42) course on the information security 2, which was basically concentrating on how to build, how to use contemporary hardware features and build secure system. So, we called it as architectural aid to information security.

So, when we look at typical systems, first there is an hardware on top of it there is an operating system and of course, the network, the computer communication network, which basically connects multiple systems also start interacting at this point. So at the third level of this information security series, we will present you the basics of UNIX and network administration with some emphasis on security. So this will form the basis for the information security 4 course and the subsequent courses in the series, which will basically a talk about security at higher layers above the operating system and applications etcetera.

(Refer Slide Time: 1:42)



2

## How is this Course Organized

- Lectures through MOOC
- 60 Modules of 20 minutes each
- 8 (MCQ and/or programming assignments)
- Basic Programming language
- End course exam

Operating Systems

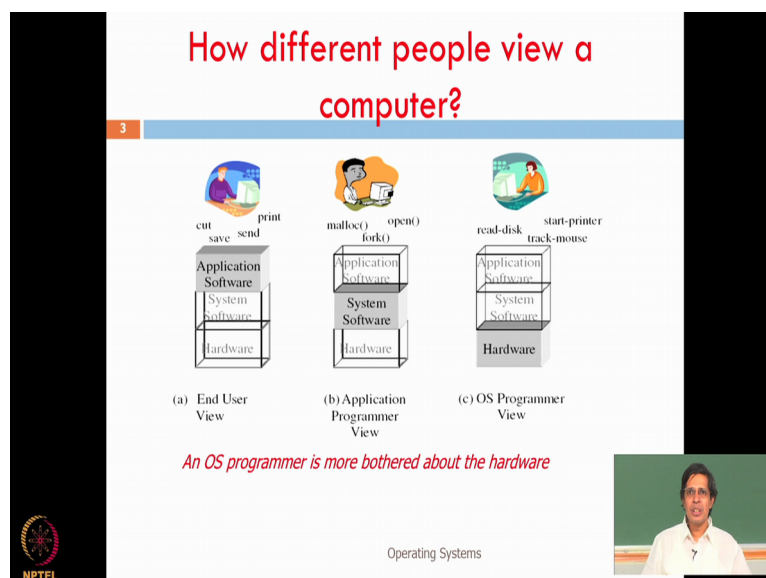
NPTEL

Operating Systems

Operating Systems

So, this course will be having 60 modules of 20 minutes each spawning (1:46) 20 hours. We will have 8 programming assignments of both MCQ multiple choice questions and programming nature and then actually assume that you have some basics programming exposure (1:59) to a basic programming language and of course , they all be in end course exam and you have to clear both the end course exam and the programming assignments to basically get the certification right.

(Refer Slide Time: 2:11)



3

## How different people view a computer?

cut save send print      malloc() open() fork()      read-disk start-printer track-mouse

Application Software  
System Software  
Hardware

(a) End User View      (b) Application Programmer View      (c) OS Programmer View

*An OS programmer is more bothered about the hardware*

Operating Systems

NPTEL

Operating Systems

Operating Systems

So we go into the module one which basically introduces some of the fundamentals of operating system. So when you look at a system, there are three different views, one is the end user, who uses the applications provided to him, for example, I use a browser yes I use a word processor, I am an end user and what do I use? I use an application software that is provided to me and who writes this application software? An application programmer actually develops this application software. So the typical functionalities that I use as an end user instead of (2:45) I use the for example, I use a word editor and I say cut the text, I save the file, I send the file, I print the file etcetera. So I do not really bother how the software actually works, I use the functionalities offered by the software, I am the end user and I do not know anything about the system, I just know how to use the software namely a word editor for example. So this is the end user view.

Now, somebody has written the software for you and that person is the application programmer and what is the application programmer wants? He actually has to develop the application using some of the features available to him and who gives those features? A system software actually gives these features to the application programmer to make him write the application, for example, when I am developing an application, which will be used by some end user. Let us assume that I am developing the application using C programming language. So what do I require, I require memory. So as a C programmer I will ask for malloc. Malloc is that when I am executing a program I need memory as a application software developer, I will ask the operating system or the system software, give me some memory. This is called dynamic memory allocation you can go and look at any book in C programming language and understand what is malloc.

So malloc is an example of some feature, which the application software would require from the operating system to basically do it is computing. Similarly, I want to open a file, I want to printf at the on the screen printf, scanf many many features that an application software uses. These are all supported by some other software layer below it and that is called the system software. So from an application programmer view, he bothers about what the system can offer to him and basically that is what we call as the system software.

Now, somebody has to develop the system software and that person is basically whom we call as a operating system programmer or the system programmer. So we have an end user, who does not do any programming, he just uses the software. The end user uses an

application that application is developed by an application programmer who actually uses the facilities given by a system software to develop the application.

Now, somebody has to write the system software and that person is a systems a programmer and or O.S. programmer and he needs to know about how the hardware actually could be used, for example, he reads know how to read from a disc, he needs to know how to start a printer, he needs to know how to track a mouse. So there are three different views an end user view, an application programmer view and the OS programmer view. So a system programmer is mother bothered about the hardware. So in our last course information security two course we have actually talked about what are the interfaces that hardware can give to your system software developer. Now, in this course we will see how the system software developer can utilize the interface that is provided by the hardware. So, this is how we linked these three courses.

(Refer Slide Time: 6:16)

**System Software**

- Independent of individual applications, but common to all of them
- Examples
  - C library functions
  - A window system
  - A database management system
  - Resource management functions
  - The OS

One can say an OS is just a software but let us see how misleading it can be though the statement is true.

Operating Systems

The diagram illustrates the layers of system software. At the top is the 'Application Programmer' who interacts with the 'Software API'. Below this is the 'System Software' layer, which includes the 'Command Line Interpreter', 'Loader', 'Window System', 'Compiler', 'Libraries', 'Database Management System', and 'OS'. These components interact with the 'Hardware' at the bottom. A small video inset shows a man speaking.

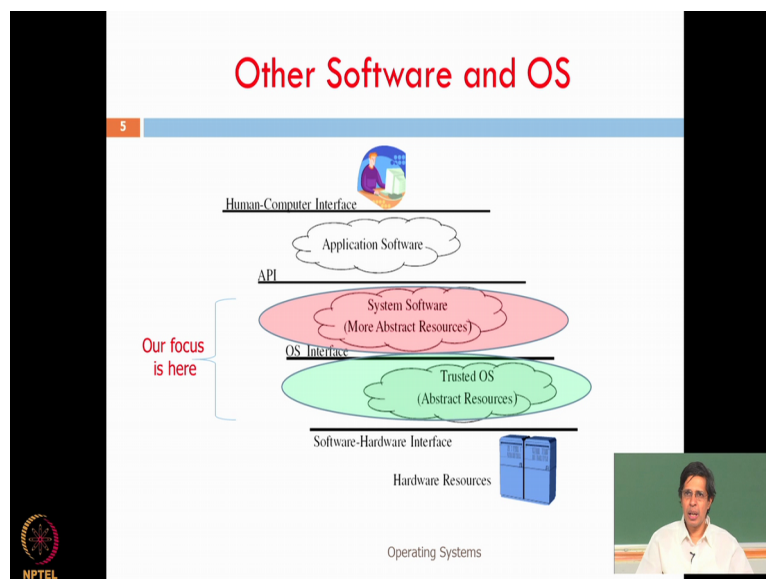
Now, what is system software? System software for example, when you write a C program, you use lot of headers like hash include, studio dot h, stdlip dot h, math dot h, string dot h. So lot of facilities are given to you. So a bunch of what are these dot h files, they are header files, which basically point to some libraries, which are prebuilt and you can use it for doing certain functionalities, for example, if you do not have studio dot h, if Suppose you have to write your printf all by yourself, you will actually spent the four years if you are an undergraduate student, you will spent the rest (6:55) of your four years just to write your hallo world program, because printf is such a complicated function, you need to understand lot to implement printf, but you finish your hallo world program in to save, the first five

minutes of your introduction to computing course. How do you do it, because you used printf, which is given by somebody else and who is that the system software.

So a system software, a collection of C library functions can be called as a system software. So, when you logging to your system, you basically have a window, a dextop coming up that itself your system software. Now, when you are developing a web application and you actually basically use a database to save for example, store all your login and password or to store some of your credentials your data. So for example, my SQL is a database management system. So DBMS can it is system software. Then there are lot of resource management functions coming through the form of device drivers etcetera. So this is also we can say (( )) (7:57), they are all system software and of course, finally the operating system.

One can say that then operating system is just a software, but actually it is more than a software and we will see though, it is a software that is executing, but the general definition say, we cannot say that an operating system is actually, a bunch of library functions like what a C library function. We cannot compare the four blue points that I have given as example with the red point that is the OS there. So for example, a OS cannot be compared with a C library function, it is also both are software, but they are something different and let us see what it is as we proceed in this module. Now, all these system software is available to the application programmer through what we call as an application programmer interface API a software API and this system software basically as I mentioned in the previous slide, we will execute on the hardware.

(Refer Slide Time: 8:57)



Now, in this course we will bother more about system software. So as you see the entire stack, stack is one over another at the bottom you have hardware resources and this hardware is basically utilized by the software to execute. So there is an interface between software and the hardware. This interface we covered in great detail in information security two course, which is actually available for all of you to view what we call as architectural aid to information security. If you have not done that course do not worry, this information security three course is independent or is self-content of the information security two course, but if we actually done that you will have a better appreciation of this course nevertheless (9:40), you can safely do this course without attending the information security two course, you try and explain lot of things and bridge those gaps that much I can assure you here.

Now, the hardware actually offer certain interface to the software and so, there are two parts to the system software, one is a trusted OS, which actually gives which abstracts out the resources to the higher layer right. The trusted OS sometimes we call it as kernel. So what is functionality of this? So a person who is writing a system software who is above this layer, for example, pink layer we are now talking about the green layer. Thus programmer who is working at the pink layer need not really bother about the hardware resources.

So this kernel basically abstract these resources and gives this as an interface to the pink layer. The pink layer basically is a system software layer, which is still abstract it more and so when an as an application software when I am developing it, I really need not bother what the hardware underlying hardware resources, I can just write, for example, when you write a printf statement, do you really bother what is the monitor there, no you can have any variety of monitor, you just say printf, somebody takes care of taking your printf and printing it on say, for example a monitor made by company X or monitor made by company Y, you say fprintf, do you really bother what is the hard disc that is stored there, you do not bother.

So it can be by some company X or Y somebody else takes care of taking your fprintf and actually saving it, in your file if you say fopen, open the file, fclose close the file, I am taking examples from C programming language just to explain things better. So all these things are taken care of by some layer and that layer is the operating system come system software. So we will be focusing on the pink and the green that we are talking of in the screen, right and so, course number two actually covered a (11:53) about hardware resources specifically

from information security perspective. Now, we will cover lot about the two layers here and then subsequent courses we will talk about application software.

(Refer Slide Time: 12:07)

The slide, titled "Components of a Computer", features a list of components on the left and a layered architecture diagram on the right. The list includes:

- Hardware – basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games

The diagram illustrates a four-layer architecture. At the base is "computer hardware". Above it is the "operating system". The next layer is "system and application programs", which includes "compiler", "assembler", "text editor", and "database system". At the top, multiple "user" boxes (user 1, user 2, user 3, ..., user n) are shown, with arrows indicating their interaction with the application programs layer. The NPTEL logo is visible in the bottom left corner, and a small video inset of the presenter is in the bottom right.

Now if you look at the components of a computer. So we basically have a computer hardware on top it, executes the operating system and top of the operating system we have system and application programs. The system and application program basically compresses the compiler, your assembler, your text editor, data base system and then on top of this, there are the users, who uses all these things. So there can be a user uses the compiler somebody can use your text editor, somebody can use your web browser, somebody can play videogames on top of you etcetera.

So there is a four layer when we view the system, there are four layers, the basic is the computer hardware on top of it is your operating system, then your system and application programs and then the end user, who uses this application program. To again say, in this course we will talk more about the operating system and also, the network which actually forms a major the interaction of a communication network to a system basically is through the operating system. So we thought that we could make this together and give you a holistic view of how an operating system and network work together.



(Refer Slide Time: 13:22)

## Components in Action



7

- An operating system is a crucial link to put the all the computer "components" in action

The diagram illustrates the interaction between four layers of a computer system:

- User:** Provides **Input** to the Application Software and receives **Output** from it.
- Application Software:** Sends **Service Requests** to the Operating System and receives **Service Responses** from it.
- Operating System:** Sends **Hardware Instructions** to the Computer Hardware and receives **Processing Results** from it.

Operating Systems



Now, this typically the interface between the various components of a computer. So as a user you actually drive and input to an application software, for example, you give a C program and you compile it and when the program executes, it actually request for some service from the operating system for example, malloc please give me some memory or say printf, printf on the screen.

Now, the operating system will take this service request and need to get it executed on your hardware. So basically this operating system translates your service request into a set of instructions that will be understood by the hardware, for example, we say printf somethings, the operating system will take that and through some means through a device driver, it will go and tell to the screen to their graphics processing path of your system, go and print this. So it basically gives hardware instructions and in return your computer hardware will say, in the case of printing it just prints, but in the case of say, read something from the hard disc. So it actually reads and it gives back a result of what it has read and once, the operating system gets the data then it will give back the application software as requested something, for example, it requested some memory, operating system will actually allocate that memory and then give a pointer to that memory that is what we mean by service response and this will basically we given as an output to the user.

So the user ask for certain things to the application software, which in turn ask for some services from the operating system, which again ask which translates into set of hardware instructions get it executed on the hardware, the results obtain back to the operating system, which in turn gives response to the service request made by the application software, which



in turn is converted into an output and the end user can view it. So this is basically how the four components actually interact.

(Refer Slide Time: 15:18)

The slide is titled "So a few things become obvious..." in red text. It contains a bulleted list of questions and answers about the OS. Below the list is a diagram showing the layers of an operating system: User programs, Operating system interface, Operating system, Hardware interface, and Hardware. The NPTEL logo is in the bottom left, and a small video inset of the speaker is in the bottom right.

- What is an OS?
  - Is a program between computer users and computer hardware
- Why do we need OS?
  - We need an environment in which a user can execute programs.
  - The purpose is three fold:
    - To ensure the correct operation of the computer system
    - To make the computer system convenient to use.
    - To use the computer hardware in an efficient manner.

Operating Systems

Now, in this whole thing what is this OS? It is nothing but a program between users and the computer hardware, right and why do we need an OS we need a OS for lot of things. One of the important things that we need to talk off and that is also very important from an information security perspective is that, it abstracts the complete hardware from the user. So as I mentioned you, when I want to do a printf, I never care what is the actual monitor or what is the hardware that is executing there, I just tell printf and some layer takes care of taking this printf and getting it printed on the screen and so, this is very very important.

So one of the most important obvious use of an operating system is to see that there is an abstraction and because of this abstraction, your computing becomes very convenient, right and since, the operating system is taking care of lot of co-ordination you can start using your hardware in a very efficiently manner and also operations would be well tested by the operating system. So if you request operating system for an operation then the operating system also ensures that it does the operation correctly, for example, if I want to read a file I want to read some bytes of the file, you just tell operating system, this is what I need, the exact that bytes will be read and given back. So the operating system is also responsible to see that your operation is correctly executed. So these are some need for an operating system.

(Refer Slide Time: 16:51)

The slide is titled "Strategies of OS => Different types of OS" in red text. It features a list of OS types and their characteristics, organized into two columns. The left column lists various OS types, and the right column lists their characteristics. A small video inset in the bottom right corner shows a man speaking. The NPTEL logo is in the bottom left, and "Operating Systems" is at the bottom center.

- Depending on the end application, hardware.. so on, O.S has different strategies to do its role.
- This can be conveniently referred as "different type of OS"
  - ▣ Batch processing
  - ▣ Timesharing
  - ▣ Personal computer & workstations
  - ▣ Process control & real-time
  - ▣ Network
  - ▣ Distributed
  - ▣ Small computers
- This can be grouped based on (and also named) as follows
  - ▣ Single user
  - ▣ Multi user
  - ▣ Single tasking
  - ▣ Multi tasking
  - ▣ (or) Single programming and multi programming.
  - ▣ Distributed
  - ▣ Embedded
  - ▣ Real time

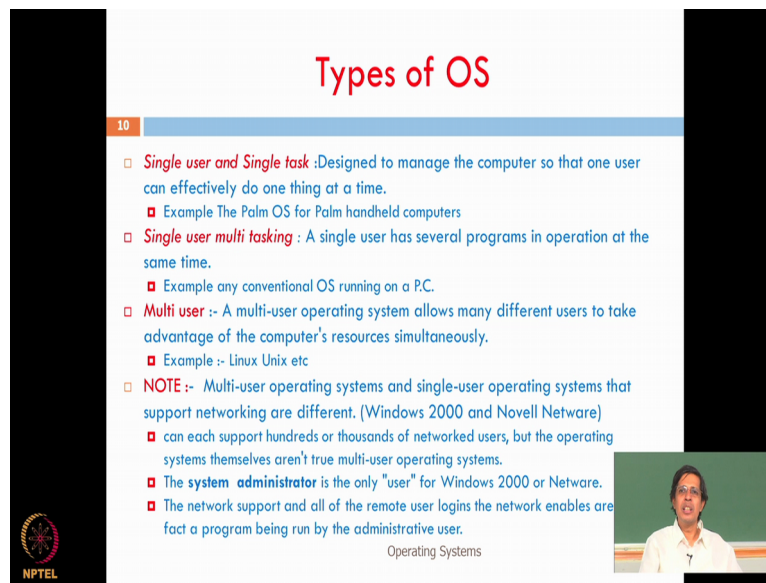
Operating Systems

So very quickly there are different types of operating system that we see even today. The type of operating system actually depends upon the final end application, which you intend to use. So there are several types of operating system very quickly, you could have a batch processing operating system, a time shearing, a personal computer and workstation based operating system, a real time operating system, a network operating system, which run runs on your routers and your switches today, your network appliances today, a distributed operating system, operating systems for small computers like embedded systems.

So different type of OS are there depending upon how your end application is, we will just see more detail definition of this very shortly and then of course, the OS can also be grouped depending upon other criteria like for example, single user operating system, there will be only one user, who uses that system like for example, your dextop operate you may be the only user, who is using it, you may not shear it with across you different users.

So there are lot of both execution and the security implications by saying that this is a single user operating system verses a multiuser operating system, for example, your mail server many users can login. So it is an the operating system that running's on your mail server should be a multiuser operating system, we will have single task operating system, multitasking operating system, single programming and multiprogramming operating system we can talk that. There is a distributed operating system, embedded operating system, real-time operating system. So these are all different types of operating systems that we see. So very quickly we will define these things, because these are very important for us to further investigate on this.

(Refer Slide Time: 18:33)



The slide is titled "Types of OS" in red text at the top center. Below the title is a blue horizontal bar with the number "10" on the left. The main content is a list of bullet points, each starting with a red square icon. The first bullet point is "Single user and Single task", followed by "Single user multi tasking", "Multi user", and "NOTE". Each bullet point includes a brief description and an example. At the bottom right of the slide, there is a small video inset showing a man in a white shirt speaking. The NPTEL logo is in the bottom left corner, and the text "Operating Systems" is at the bottom center.

- **Single user and Single task** :Designed to manage the computer so that one user can effectively do one thing at a time.
  - Example The Palm OS for Palm handheld computers
- **Single user multi tasking** : A single user has several programs in operation at the same time.
  - Example any conventional OS running on a P.C.
- **Multi user** :- A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously.
  - Example :- Linux Unix etc
- **NOTE** :- Multi-user operating systems and single-user operating systems that support networking are different. (Windows 2000 and Novell Netware)
  - can each support hundreds or thousands of networked users, but the operating systems themselves aren't true multi-user operating systems.
  - The **system administrator** is the only "user" for Windows 2000 or Netware.
  - The network support and all of the remote user logins the network enables are fact a program being run by the administrative user.

Operating Systems

A single user operating system or single task operating system is design to manage the computer so that one user can effectively do one thing at a time. A single user multitasking is a single user has several programs in operation at the same time, for example any conventional OS like your Linux or windows that is running on. A multiuser operating system allows many different users to take advantage of a computer resource simultaneously, for example, your LINUX Unix servers many people can login; a mail server is an example of a multiuser operating system.

Now, there are operating systems which are basically done for networking purposes. So this is one certain difference that we want to see a multiuser operating system and a single-user operating system that supports networking. So there are two things, a multi-user operating system, a single user operating system that support networking, they are actually different, for example, windows 2000 and Netware, these are actually single user operating system. There is only a system administrator, but it can be used to actually networks thousands of systems, it can each supports hundreds or thousands of networked users but the operating system themselves are not truly multi operating systems.

So we are just trying to classify this we also suggest that you go and do a little more reading about windows 2000 and Novell Netware through Google or Wikipedia to get more inside into that, but we are trying to classify operating systems, because the understanding of this type is extremely crucial for us to go further.

(Refer Slide Time: 20:05)

The slide is titled "Types of OS" in red text. It contains a list of three types of operating systems, each with a brief description and examples. The first is "Real Time Operating System", described as managing resources for precise execution time, with examples like aircraft control systems. The second is "Distributed Operating System", which manages multiple independent computers as a single one. The third is "Embedded Operating System", designed for small computers with limited resources, with examples like Windows CE and Minix 3. The slide also features the NPTEL logo in the bottom left, the text "Operating Systems" at the bottom center, and a small video inset of a speaker in the bottom right.

- **Real Time Operating System** manages the resources of the computer so that a particular operation executes in precisely the same amount of time, every time it occurs.
  - Operating system in air-craft or a control systems
- **Distributed Operating System** Manages a group of independent computers and makes them appear to be a single computer.
- **Embedded Operating System** Designed to operate on small computers with a limited number of resources.
  - They are very compact and extremely efficient by design.
  - Windows CE and Minix 3

In there something call real-time systems, for example, a real-time system is what you expect out of an operating system we say that if I ask for an operation X, I should get a correct result. This is a normal operating system, but in a real time operating system, when I ask for an operation X that the operation should be done correctly not only it should be done correctly, but it should be done within a given point of time, for example, I am running an aircraft or a any control system, let us say aircraft, I put the landing gear or something like that I do not know how aircraft works, but let us say, I want to put a landing gear and which will actually bring down the wheels so that the plane can land.

Now, the wheels should come down within say, next two minutes suppose it takes (20:48) something happens to the operating system and takes one hour for the wheel to come down, the plane cannot even land and it may go out of air (20:55). So the operating system that actually controls an of aircraft should ensure that the operation is correctly done, but not only that it is correctly done, but it should also be done within a particular time interval. So that is a definition of a real-time operating system and there are distributed operating system, which actually manages a group of independent computers and make them appear as a single computer from your application software perspective and there are embedded operating system that you see everyone has a mobile today right.

So what runs there can be classified as embedded operating system or a real-time operating system, it is (21:33) a very thin layer, but suppose you look at an internet of things IOT devices. If you do not know what is IOT please, go to Google and find out what is IOT, but these are very very small devices, which are used to find what is the pressure, what is a

temperature in a room, for example, something that runs on your washing machines, some fussy logic that is running on your washing machine. So there are all examples of your embedded operating systems and these are very very compact and extremely efficient by the science (0)(22:02).

(Refer Slide Time: 22:17)

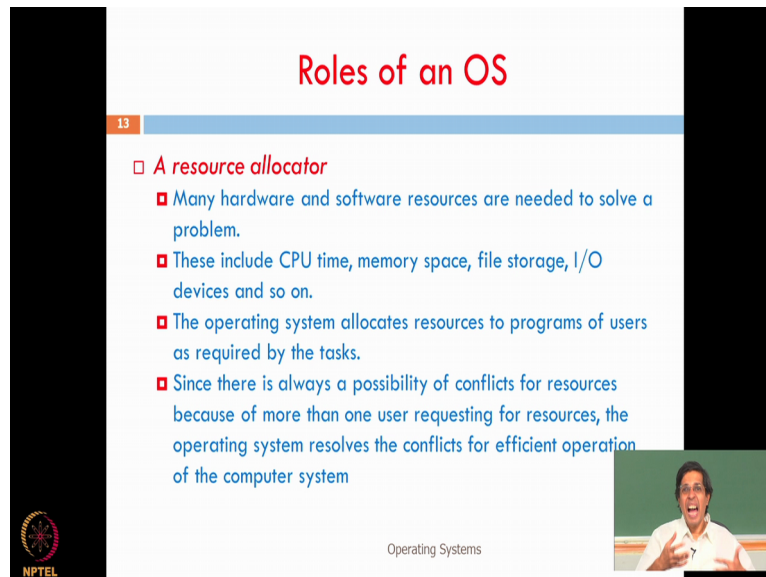
The slide, titled "OS Evolution and Grouping", is slide number 12. It features two main diagrams. The left diagram is a timeline from 1960 to 2000, showing the evolution of operating systems across different computer types: mainframes, minicomputers, desktop computers, and handheld computers. It lists various OS types like batch, time-sharing, real-time, interactive, and networked, and their associated architectures like single processor, multiprocessor, and distributed systems. The right diagram is a conceptual model of an operating system, showing "Operating Systems" at the base, which branches into "User Interface", "Resource Management", "Process Management", "File System", "Device Drivers", "Client/Server", "Network", and "Real-time".

So if you look at the evolution of OS, we have started from what we call as mainframe computers, which basically does batch processing, for example you give the job today and tomorrow morning you come and collect to mini-computers and then to desktop computers and to hand-held computers. The major transitions from mainframe to the hand-held computers is that we are now looking for interactive computing. I start doing some work and at every step I am expecting the computer to give back the results and so, this is what we mean by an interactive computing and also that these systems are now heavily networked, moment I switch on my mobile, I am connected to billions of people billions of such devices across and then we start shearing information.

So today the most important aspect of operating system both from the development of the operating system point of view from the requirement of the operating system point of view and also from the view of you know, the security of the operating system is that the system are become highly interactive, the systems are expected to be real time, in sense that we need very quick results and that the systems are heavily networked right. So these three form the basis of development of operating systems today and when we start looking at maintaining this operating system, administering this operating system. It is very important for us to understand these requirements today that we have users, we have multiple systems connected

to a network, we have users, who want to use multiple applications and who are looking for very quick results and this forms the basis of our understanding of contemporary operating systems.

(Refer Slide Time: 23:59)

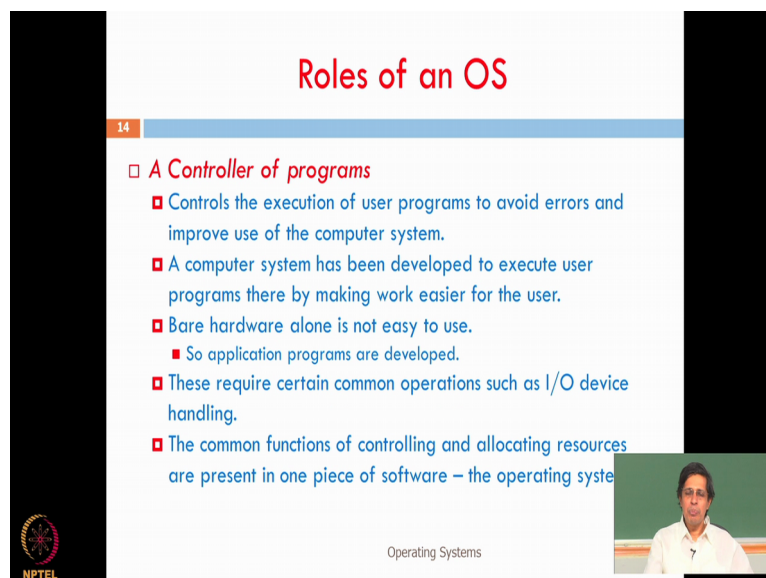



Slide 13: Roles of an OS

- **A resource allocator**
  - Many hardware and software resources are needed to solve a problem.
  - These include CPU time, memory space, file storage, I/O devices and so on.
  - The operating system allocates resources to programs of users as required by the tasks.
  - Since there is always a possibility of conflicts for resources because of more than one user requesting for resources, the operating system resolves the conflicts for efficient operation of the computer system

Operating Systems

NPTEL




Slide 14: Roles of an OS

- **A Controller of programs**
  - Controls the execution of user programs to avoid errors and improve use of the computer system.
  - A computer system has been developed to execute user programs there by making work easier for the user.
  - Bare hardware alone is not easy to use.
    - So application programs are developed.
  - These require certain common operations such as I/O device handling.
  - The common functions of controlling and allocating resources are present in one piece of software – the operating system

Operating Systems

NPTEL



So what is the role of an OS? A role of an OS is first it is a resource allocator, I ask for a resource say, for example I ask for opening a file that means what I am looking for a file storage, I want to execute my program. So, I ask the OS give me some CPU time so that you get me executed, I am asking for some memory RAM space through malloc. So, the main role of an OS is that the application software will start asking for resource and the OS has to allocate this resource to the application software and importantly it should also see to it that

there is no conflict of these resources for example, let us say in a mail server we are trying to fellows (24:40) want to print.

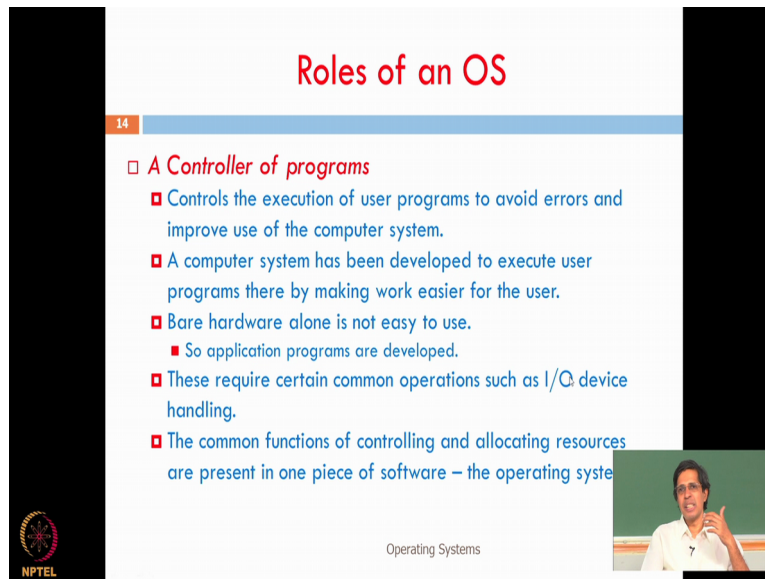
So the OS has to serialize this job, first he has to allow user1 to print then he has to bring in user 2 to print it. If both of them start printing then one line of your file will be printed next line of the file will be from his input, right. So there is a need to resolve conflicts here. So two fellows, who want to use the printer, I want to print two files, I will print file one and then file two. So this act of serialization is what we call as conflict resolution and that is one very very important role of an operating system and the operating system also actually controls the program execution for example, if you might have seen core dump, segment fault, segfaults, what does it mean? The operating system has given you some memory and if you start overshooting that memory immediately the operating system has to stop you, right.

So this basically ensure typically in a multi-user operating system where many users login and many process get executed. One process should not interfere with another process. What is a process? A process is a program in execution. So when your program is executing, some others programs error should not affect you and your error should not affect some other program, because both of them are executing in a timeshared way in the system.

So one of the important use of an operating system is to see that typically, in a multi-user operating system like your mail server, the mistake done by one program should not affect the execution of another program and as a program, when I do a mistake for example, I do a divide by zero, somebody has to stop me and say you have done a divide by zero, so you are not valid and you have to shut down my program, because it has done something, which is mathematically wrong or which is a programmatically or computationally wrong. So such type of error reporting, catching errors are also a very important role of an operating system.



(Refer Slide Time: 26:42)




Slide 14: Roles of an OS

- **A Controller of programs**
  - Controls the execution of user programs to avoid errors and improve use of the computer system.
  - A computer system has been developed to execute user programs there by making work easier for the user.
  - Bare hardware alone is not easy to use.
    - So application programs are developed.
  - These require certain common operations such as I/O device handling.
  - The common functions of controlling and allocating resources are present in one piece of software – the operating system

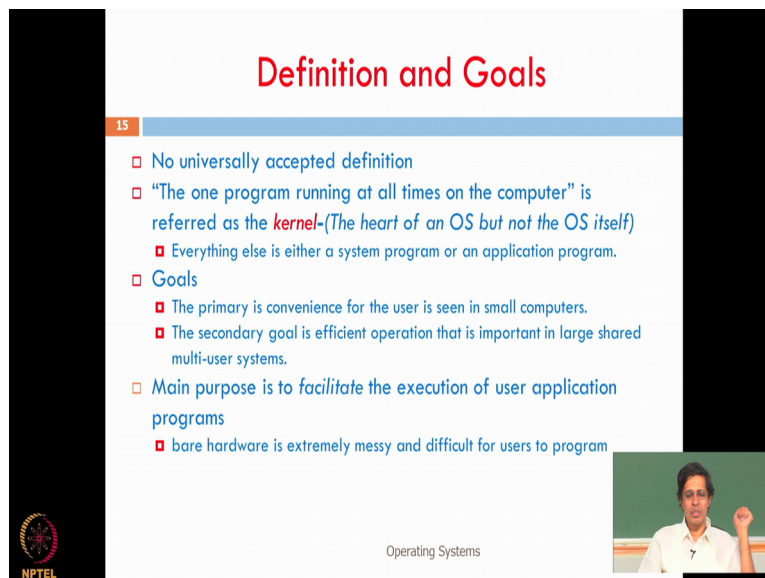
Operating Systems

NPTEL



So one of the important role of the OS in addition to allocation of resource is also to control the execution of your program and also see that your program is correct in execution and if there is something wrong then it has to stop and it has to report an error.

(Refer Slide Time: 27:01)




Slide 15: Definition and Goals

- No universally accepted definition
- "The one program running at all times on the computer" is referred as the **kernel**-(The heart of an OS but not the OS itself)
  - Everything else is either a system program or an application program.
- **Goals**
  - The primary is convenience for the user is seen in small computers.
  - The secondary goal is efficient operation that is important in large shared multi-user systems.
- **Main purpose is to facilitate the execution of user application programs**
  - bare hardware is extremely messy and difficult for users to program

Operating Systems

NPTEL



Now, what is the definition of an OS? It is so there is no universally accepted definition. The main reason is that OS is need to serve multiple users, executing varieties of programs on variety of hardware. So the OS has set of applications on top of it and it has the hardware below. So normally if we take say for example, UNIX it executes on multiple different versions of Unixes execute on multiple different varieties of hardware, right. It is not just a CPU it sees multiple devices say there is printer A, printer B, disc A, disc B.

So it is not just CPU but (it all) there are multiple CPUs also but more than the CPUs there are multiple peripherals. So, the OS sees multiple things below it and also, varieties of software. So there could be C, there could be java, C sharp, D sharp, E sharp, F sharp many many things python, HTML5, HTML4, internet explorer, Firefox, chrome varieties, open office, closed office (28:09). So you see lot of things, lot of varieties on top and lot of varieties in bottom and so, essentially your operating system is sort of (28:18) should address this variety and that is also perhaps a reason why there is no universally accepted definition for an operating system. So it is actually one program running at all times on the computer.

So moment you boot the operating system comes into existence and sometimes it is referred to as the kernel, which is nothing but a kernel is actually not the operating system it is the heart of an operating system, it is a part of an operating system. So then everything else other than the kernel can be called as system program or an application program. So it is sort of a very vague definition here but then there is one understanding, there is some basic things that execute always and we will call it as kernel and on top of it there are lot of things that basically interface the operating system with the application layer and the application software layer and the end user.

So the goals of this operating system the primary is to give the convenience, secondary goal is to ensure efficient operation, we have covered that in great detail in the previous slide and most importantly when I want execute a program, they should facilitate the execution of the as a user when I want to run an application program, it should facilitate the execution of this program.

(Refer Slide Time: 29:34)

16

## How and when does an OS start to function?

- A *bootstrap* program is loaded at power-up or reboot
  - ▣ Typically stored in ROM or EPROM, generally known as *firmware*
  - ▣ Initializes all aspects of system
  - ▣ Loads OS *kernel* and starts execution
- To understand how an OS function it is necessary to have an idea of computer organization and operation.

Operating Systems

NPTEL

So how and when does an operating system start to function? So when you just switch on your system your system will start executing from some read only memory. This is a code that is pre-stored there. So it will start fetching the instructions from a read only memory and that is those are first things that get executed and that is sometime called as firmware and what will the firmware do? The firmware will now go and look for a bootable disc, on a bootable device and it will there in that bootable device, there will be (0) as the first sector or the first block of the bootable device and it will load that program that is stored in that bootable device and it will start executing and that first program is called the boot strap program and then that will further load device kernel and give the control.

So what happens when I switch on? There is a read only memory. There are some instructions that are there that will start executing and that will basically go and find a bootable disc so that firmware will find this bootable disc and in that there will be one master boot record, whatever is there, it will disc firmware will load the content of that master bootable record into the memory and give control to this. So whatever is stored on the master boot record will start executing and that will go entered and load device kernel.

So this is how the kernel comes into existing. So right from the inception right (0) from you switch on till you switch off, there will be one program that will be always in the background in the picture and that is the operating system. So to understand how an operating system function, it is necessary to have an idea of computer organization and operation we will quickly cover that in the next module. Thank you.