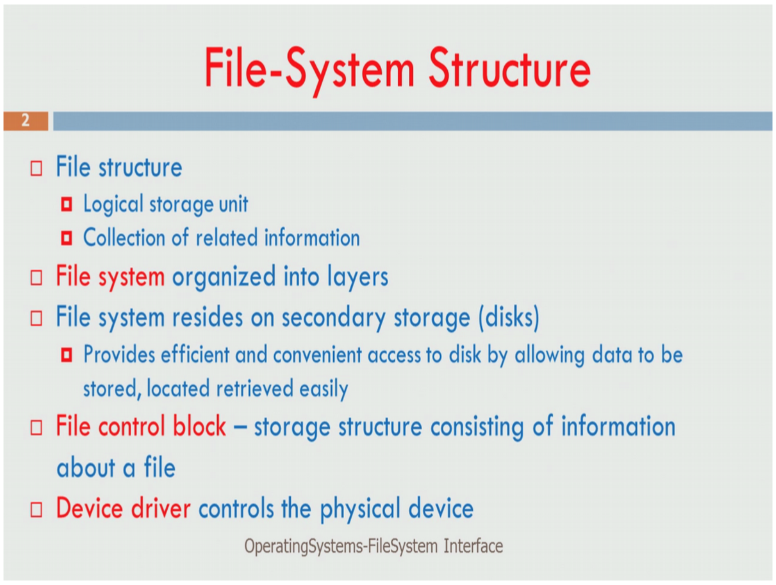
Information Security-3
Prof. V Kamakoti
Department of Computer science and Engineering
Indian Institute of Technology Madras
Basics of Unix and Network Administration
Operating Systems
Mod02 Lecture 11
Module 11: File Systems-1

So the most visible part of an operating system is the file system, right. So what you do not see anything inside, so once you open a shell or when you open your dextop what you see is files, directory is in files. So file system is very very important and files actually is the basic a file is the basic unit of storage when I even want to save one byte say one character (())(0:35) I want to just save k, I need to create a file and save that k. So file system is very very important and from a security point of view, the leakage of information happens to leakage of files.

So there are lot of things about file security of files which in the subsequent modules we will cover, right, but we will just give you the basics of how file systems are implemented so that there is a completion in which what we have trying to learn, here. So the next 2 modules I will give you some basic implementation details of file system, specifically from a unix point of view not necessarily Unix, but more on Unix, right.

(Refer Slide Time: 1:15)



File-System Structure

2

- File structure
  - Logical storage unit
  - Collection of related information
- File system organized into layers
- File system resides on secondary storage (disks)
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- File control block – storage structure consisting of information about a file
- Device driver controls the physical device

OperatingSystems-FileSystem Interface

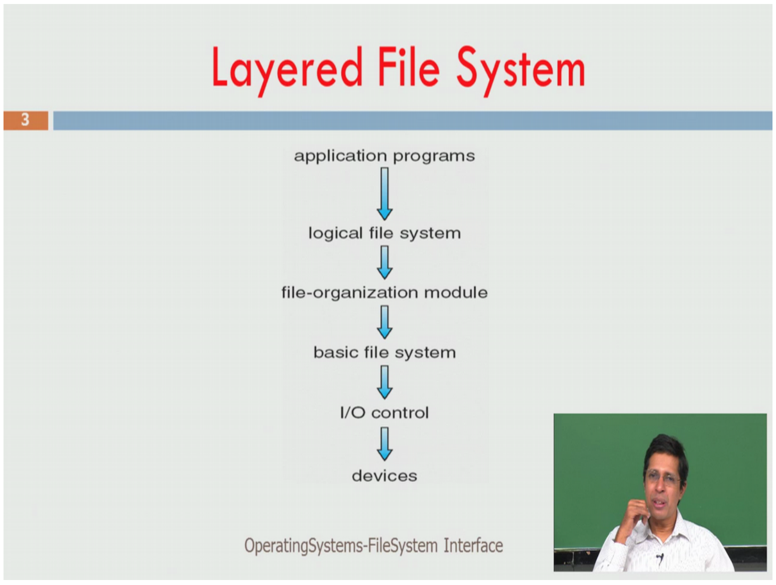
So what is a file? A file is a logical storage unit. It is just a collection of related information. These are all textbook definitions, okay. Now the file system is actually basically organized into layers, so there is again a layer like how we add a memory in hierarchy where we had disc and then we have memory, then we have cache register, like that file system itself we could have a multiple set of layers, you will get to know about these layers.

The entire file system actually recites (())(1:45) on the secondary storage and when the system boots up some part of things comes on to the main memory, right some part of the file system information comes on to the main memory and when you close down the system shut down the system this is return back, if you do not shut down the system, many part of your file system can be there in a main memory and we can continue to have it that is why if you suddenly switch off your system, then we want to reboot then it may say, file system corrupted should I fix what it means? This is what it means.

So when a system is booted up from the disc some part of the file system is basically moved into the main memory and then when the system is shut down, it is going back into the disc, so that is why you need to properly shut down your system so that next time when you boot then there is no problem with your file system. Why do I move it to the main memory as I told you main memory is much faster than the secondary memory? So to enhance the speed of your operation of your file system, some part of your disc hard disc some part of the file system, some crucial parts we will discuss some as we proceed if brought (())(2:54) into the main memory.

Now, there something called a file control block, which basically have information about the file and for every file we will have file control block and then there is a there is also one very important thing the device driver, which has to work along with the file system, the device driver is what that disc driver is, right. So your USB or whatever for every storage device there is a driver and that driver has to basically take care of the movement of data from the CPU, RAM to the disc and disc back to the RAM, okay. So these are all very very important aspects of the file system, right.

(Refer Slide Time: 3:36)



Layered File System

application programs

logical file system

file-organization module

basic file system

I/O control

devices

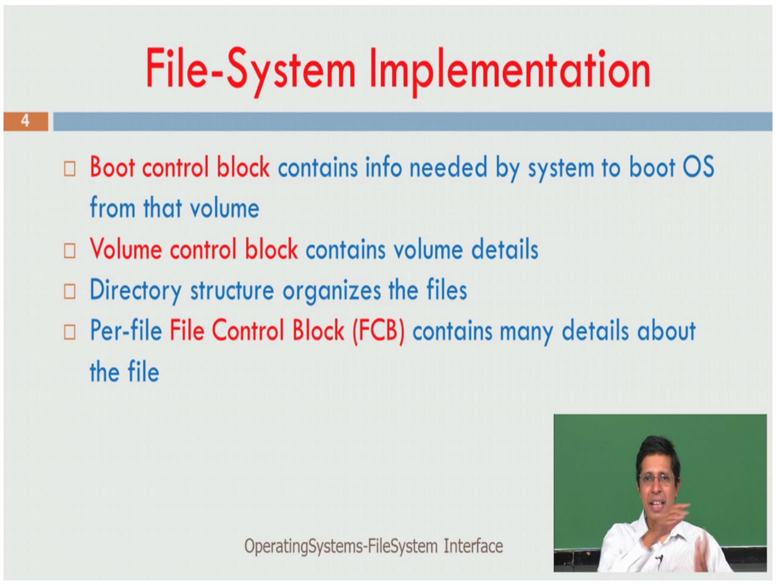
Now we said that the file system is layered. What is this layering about? So I have an application program, which wants to access a file. So it basically sees a logical file system, logical file system is that I see hello world dot C, I do not really care whether that hello world is stored on a USB disc or that hello world is stored on a CD or hello world store on a old (()) (3:59) tape or floppy or whatever I do not really care. So as an application program I see a file as a name, right hello world dot C, seating in some directory so, right. So that is what my view as an application program is a logical file system, I am not bothered about how physically that file is stored in which media and which form.

Now somebody has to hide that physical storage from me and who does it? The file system part of the operating system does it, right. So I see f logical file system. What does the logical file system see it? That see is an file organization module. What is file organization? The file is organized like a tree, right. So you have a root file system inside that root, root directory inside that root, there are several directories, inside that several directories, so it is an hierarchy of directories and in each directory I could have other sub-directories plus files. So that is how, directory itself is a file, which will give you the list of files that are stored here. So that is the file organization.

The file organization module sees a basic file system and then the basic file system basically sees an IO control with has to read and write and that the IO control basically sees the devices, right. So there is a complete abstraction at the highest layer of an application program just seeing a file as a name or a directory slash name then there is a file logical file system looking at you sees is a file organization module. File organization module sees a basic file system, the basic file system sees an IO control, IO control basically sees the devices. So this is the layered file system.

Why we need a layered file system then only application programs can be easily written. So application program need not have knowledge about the basic implementation and this is also impor the other important thing is that there could be the operating system could have much more control on the access of the files, right. So giving this abstraction, the operating system can give permissions for say user can use only this part of the file system, right. So user B can use some other part, user A and user B cannot have intersection in what they can use or they can have intersection, wherever they have an intersection, it will be only a read only files system. The user can only read from this, user cannot write it into this part of your file system. So all these type of access permissions can come up we going to talk about that in the later modules, but that is very very important the layered file system has lot of implications on security.

(Refer Slide Time: 6:41)



File-System Implementation

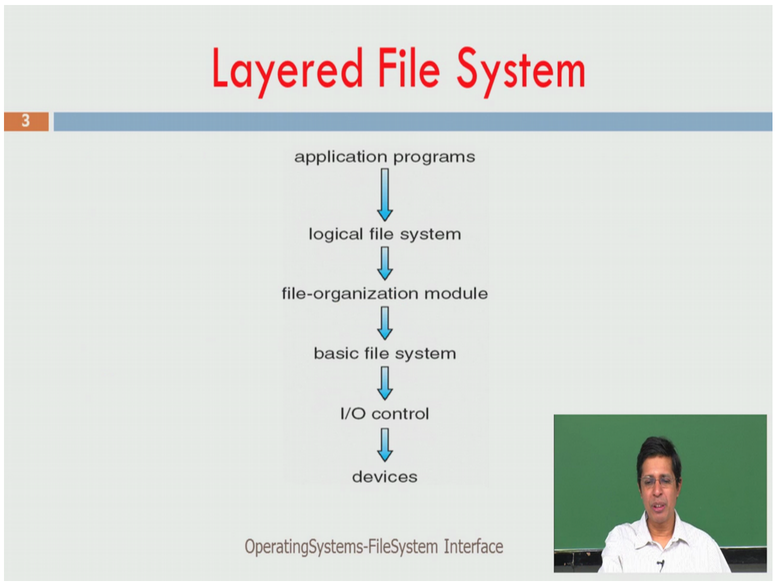
4

- Boot control block contains info needed by system to boot OS from that volume
- Volume control block contains volume details
- Directory structure organizes the files
- Per-file File Control Block (FCB) contains many details about the file

OperatingSystems-FileSystem Interface

So a file system essentially has one thing called a boot control block. So when we boot a system it says, so I am looking into a bootable device. What you mean by a bootable device? This is a disc in which the first entry or we call it as sector zero or the first that should have a bootable program. So that is called the boot control block. Then it has something called the volume control block then it has a file control black. So boot control block basically enable booting through that particular disc. The volume control block actually controls, so there could be multiple volumes and then each volume is controlled and within that volume control block we have a file control block, which basically manages the entire directory structure etcetera.

(Refer Slide Time: 7:34)

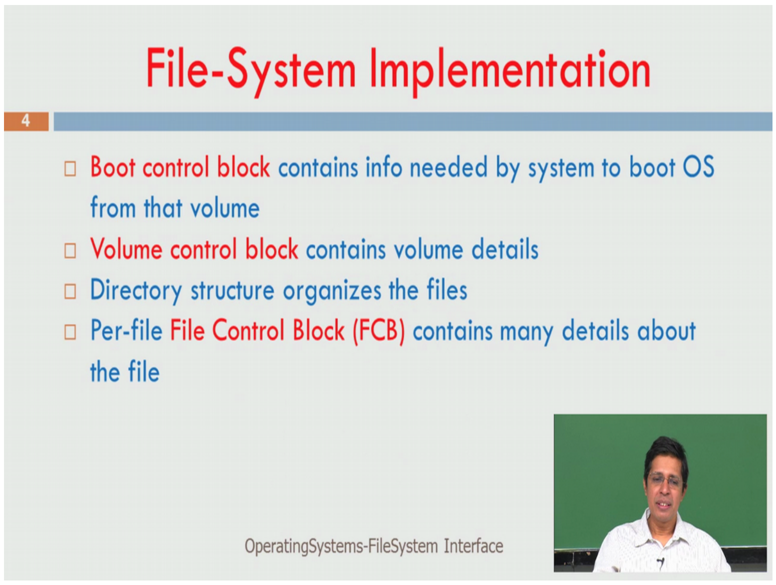


Layered File System

3

application programs
logical file system
file-organization module
basic file system
I/O control
devices

OperatingSystems-FileSystem Interface



File-System Implementation

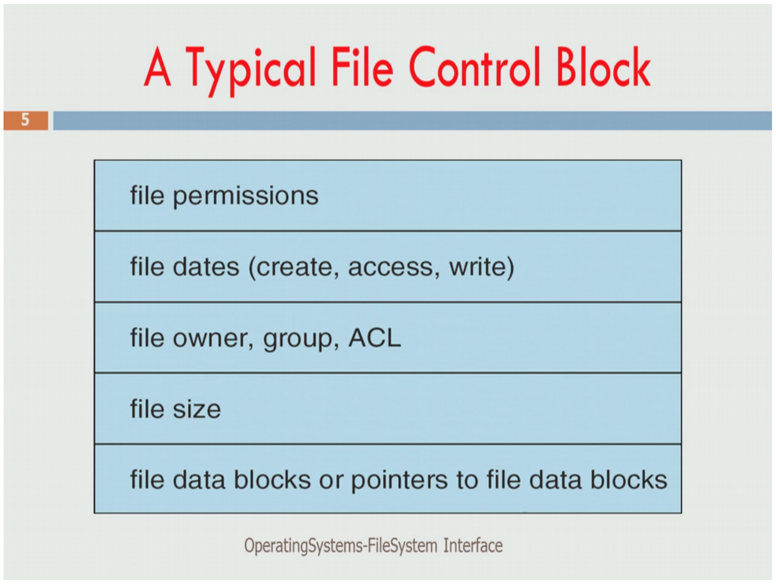
4

- Boot control block contains info needed by system to boot OS from that volume
- Volume control block contains volume details
- Directory structure organizes the files
- Per-file File Control Block (FCB) contains many details about the file

OperatingSystems-FileSystem Interface

So again there is a layer there within a basic file system itself what we call here, the basic file system you see 3 from the bottom, there could be hierarchy there or there could be a partition there of responsibilities. The boot control block is responsible for booting the system, volume control block to maintain the entire volume then file control block is for maintaining individual files. So there is an hierarchy created there. So all these hierarchies are important, because this is where we basically you know any (())(8:00) into this can cause serious information security issues.

(Refer Slide Time: 8:07)



A Typical File Control Block

5

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

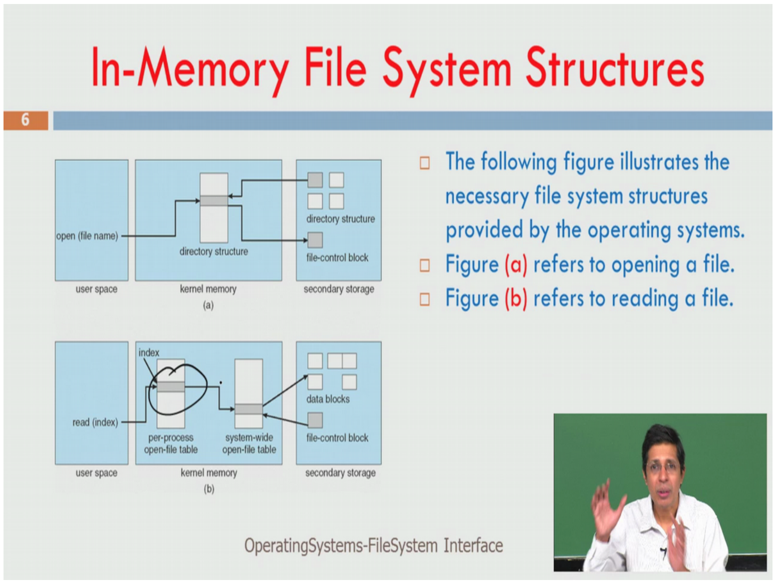
OperatingSystems-FileSystem Interface

Now we will now look at the typical file control block the bottom most part of this. So it has file permissions, who can create the file? Who can delete the file? Who can write into the file etcetera? File dates last modified, for example, I talked about in the previous module about a disc rental (())(8:27) employ going and changing a boot time variable, right. So now we could actually have when was it last change (())(8:34) so that access time etcetera.

So file dates or access times are very very important and then who is the owner of this file? Who to which group is belonging to? What is the access control list for this owner etcetera, then the size of the file is also important. Then we should have pointers of, so these are all information about the file. So these are called attributes of the file, say for example you are person I am come across to you (())(9:00), I am my initial is v, I am so and so, my age is so much, my gender is so, so I have some attributes, so like so for (())(9:07) like that the file that the file also has some attributes, after that the data of that file should be store. So it will also have pointers to where the data is basically stored, right.

So the typical file control block in the Unix environment we call it as an Inode or an identification node. It has all these details about the files and it will also have pointers of where the data of those files are stored.

(Refer Slide Time: 9:32)



In-Memory File System Structures

- The following figure illustrates the necessary file system structures provided by the operating systems.
- Figure (a) refers to opening a file.
- Figure (b) refers to reading a file.

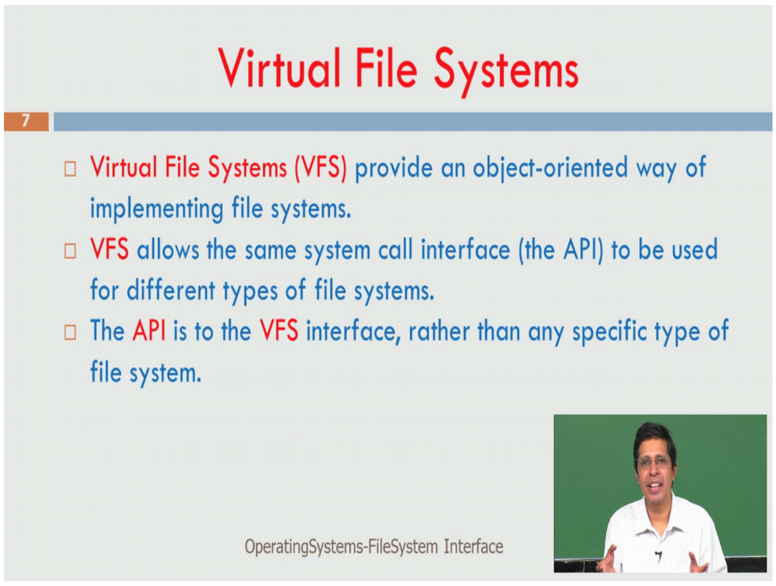
So as I mentioned there could be a In-memory file system structures. What is that In-memory file system structures? When I boot the system there should be something that is brought from the disc into the memory, typically to enhance the speed and also maintain certain security inside also, right. So what will be there inside a memory, for example, we create, a open a file, for every file there is an entry created in a file table the file table is an In-memory file structure. The file table will have details about all the files that are basically opened at that point of time which is currently being accessed by you at least one of the process that is executing in the system, right.

So for every process we have an In-memory file type that is what we call as per process open file table and then I could have one file table for the entire system system wise. So these are all basically important things that are brought into the memory so that we could we could handle this. So what will be there inside an entry in the open file table as we see here? So what could be there?

So when I start accessing a file please note that say, let us say a C program I say fscanf, so I read byte after byte. So one of the important, so I do fscanf, now I read some byte and I do some other processing and some later point I do an fscanf I read exactly the next byte after what I have read. So this open process file table will keep some information about how many bytes are currently read so that when the when I want to read the next fscanf will actually get me the next byte from that.

So it will keep information about how many bytes are currently read, right. I can also open a open a file with read write permission, so if you look at a fopen in your C code, you can open it with read or write or read-write permissions that permissions also will be put here. So when the file system comes into existence, there are lot of things that are brought into the main memory for so that we can have quick access and also we can manage the operations on this, so that is what we mean by In-memory file system structures.

(Refer Slide Time: 11:50)



Virtual File Systems

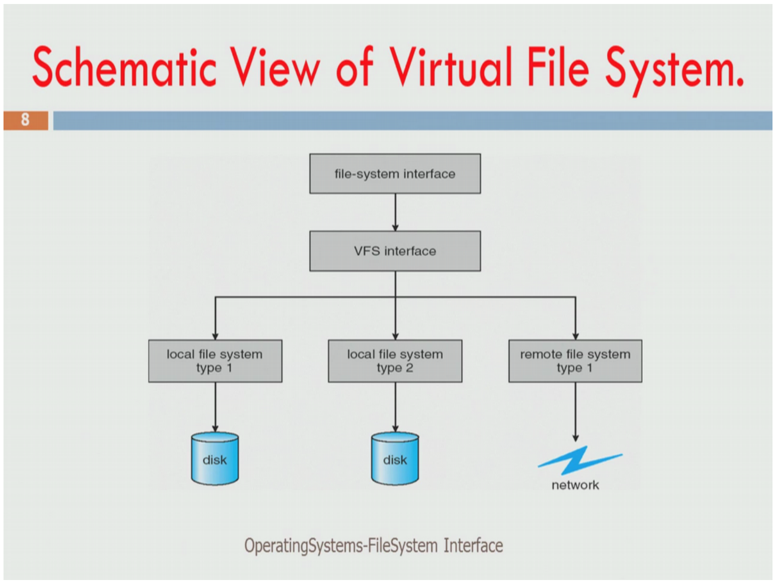
7

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

OperatingSystems-FileSystem Interface

Now let us look at what we mean by a virtual file system? The virtual file system actually provides an object oriented way implementing file systems. So VFS allows the same system call or whatever API the application programing interface to be used for different types of file systems. So that is very very important. So today I use a USB, I use a CD, I use a SD card, I use flash, I use different varieties of disc for everything, a C program whether I am opening it on a SSD or I am opening anywhere, it only says fopen it only says fopen, it only says fwrite if only fscanf, fprintf if close, that is single interface. It is independent of where you store this file. So as far as that one of the important aspect of operating system is it should review what I just explained earlier a virtual file system where you see a file as a file directory as a directory we do not really bother about how this implement and that is also very important.

(Refer Slide Time: 12:57)



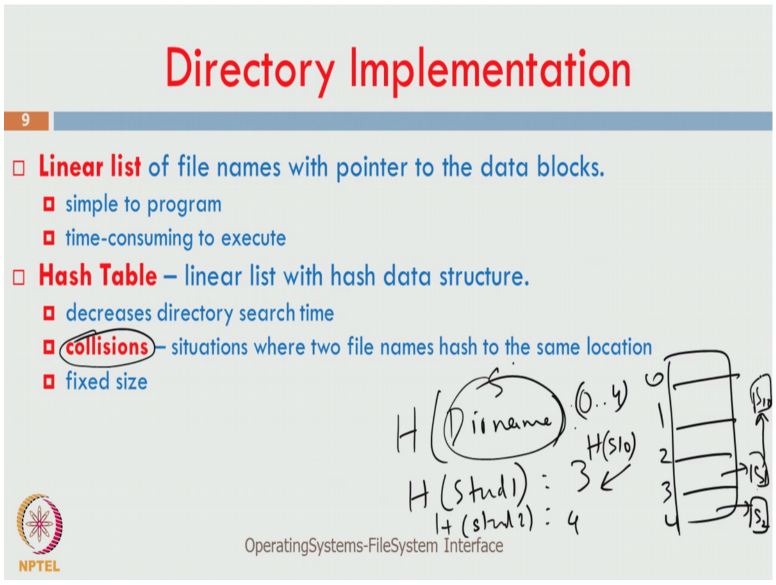
Schematic View of Virtual File System.

OperatingSystems-FileSystem Interface

So this is a hierarchy, here I have a file system interface, I have a virtual file system interface and then there could be different types, file I could have a disc, hard disc, I could have another variety of hard disc, it can be a USB or SD card and then, I could also have file system stored on some other server and this I can use using NFS or network file system, right. So **so** so the file system can reside on the network also for us if you look at typically large scale clustrures (())(13:30), right. There discs are attached to the network and there is a disc driver or a file system implemented on the disc, which basically serves the remaining compute nodes through the network, so basically giving as a network file system.

So if you want to understand more about network file system, just go and search for NAS (())(13:54) network adapter storage NAS and you can basically get more details about how file systems can be implemented through the network.

(Refer Slide Time: 14:06)



Directory Implementation

9

- **Linear list** of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- **Hash Table** – linear list with hash data structure.
  - decreases directory search time
  - collisions – situations where two file names hash to the same location
  - fixed size

OperatingSystems-FileSystem Interface

H (Dirname)
(0..4)
H(S10)
H (Stud1) : 3
H (stud2) : 4
0
1
2
3
4
S1
S2

Now let us quickly look at directory implementation. What is a directory? A directory essentially consist of list of files that are stored plus pointers to the next the directory (()) (14:14) other sub-directories. So I can implement a directory as a linear list when I am implement it as a linear list what happens? When I want to open a file, what will the operating system do? It will go to the directory and scan one by one. So if there are say 50 files or 100 files there then I want access the 100 file, I have to scan from the top and go to the 100th file and then basically pick it up.

So that time required to open a file becomes extremely, so these are very important interesting things about directory implementation. The next thing is hash table, right, so what we can do is that a given a name you would execute a function called a hash function and that will tell you an entry where it will be. So a very simple example of hash hashing, so **so** let us say that I will do some H of the directory name, right and so I have a table say 0 1, 2, 3, 4, this H will take a directory name as input and give an answer in the range 0 to 4.

So let me say H of student one, this is file that will go to say 3, so student one let me call it S1 is stored here. Now, H of student 2 will go to say 4, so S2 will be stored here. So whenever I search first S2 I again we execute this hash function and that will give me 4, so immediately I can go to 4 and get S2, right. Similarly, if I want to search for S1 I execute this H on that directory name and that will give me 3, so if I go to 3 I will get that S1. So this is so in constant time I can go to this thing, but the path is there could be this hash function might be such that there can be some H of s10 that could also give me 3. So that is what we call it as collision here. So here S1 then I will have S10 also showed.

So when I want to search for S10, I evaluate this hash function that gives me 3, I go to this 3 first see S1 then I see S10. So it will take some time it need not be necessarily constant time. So this is another very important thing. So some of these very interesting, because many of the virus you see, they actually create they go and sit (())(16:54) they can hide in the file system, basically they get themselves you know implanted inside the file system and they adjust the permissions, they can adjust the structures such that they are hidden.

So lot of these type of implementation details are exploited to gain stealth a virus comes into a disc. It is the main imp thing is stealth, it does not announce or it does not given any pointer by which you can quickly find out that it has infected the disc and for doing all these things, understanding of this directory implementation is extremely crucial that is why we are trying to cover this. These are all basic operating system fundas, but we are trying to cover this here, because you know these will also have some implications on the way a virus can affect your disc. So that is why you just virus can come and stay in the disc and with (())(17:50) stealth, okay. So that is why we trying to issue this (())(17:52).

So now a directory could be implemented as a linear list. It could be implemented as a hash table and in the case of hash table, there will be collisions and there ways of resolving collisions, one thing is to have a in a sub list for every entry as I have shown here, okay. So we will next go into the next module and talk more about file system implementation layer. Thank you.