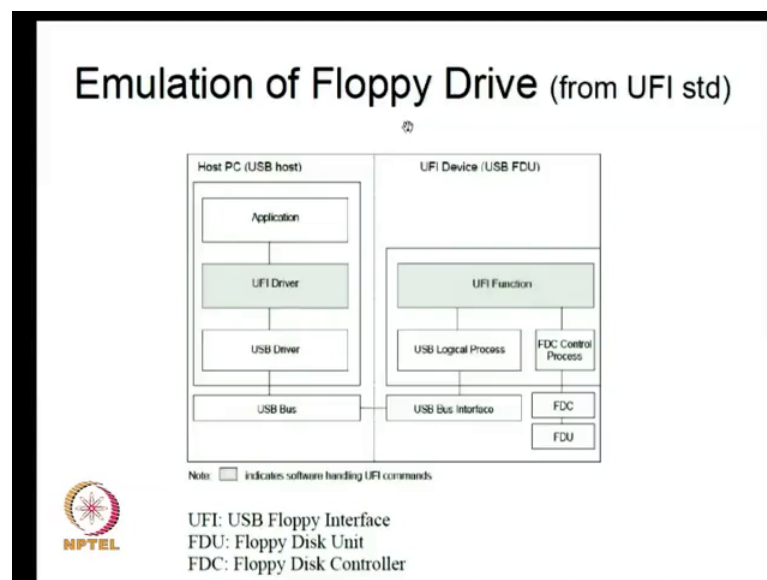


Storage Systems
Dr. K. Gopinath
Department of Computer Science and Engineering
Indian Institute of Science, Bangalore

Types of Storage Devices and Systems, Long-term Storage / Storage Interfaces and Device Drivers
Lecture - 15
Long-term Storage, Document Preservation, DNA Storage, Introduction to Interfaces to a Storage system

Welcome again to the NPTEL course on storage systems. In the previous class, we are looking at some aspects relating to how to store data that survives a long period of time, we are trying to understand what problems are, right?

(Refer Slide Time: 00:42)




I mentioned that there are issues with regarding to evolution of devices themselves you have to emulate them.

(Refer Slide Time: 00:46)

Indus Script

Yet undeciphered: meaning across time not yet accomplished
Compare with hieroglyphics (Egyptian Rosetta stone): three scripts side by side

What is the problem? Not enough contextual info:
can see the script (human "readable")
no mapping between symbols and phonemes
need interpretation of sequences of symbols
a problem in archeology, history, society,...




You also look at try to look at something closer to what happened in this country. For example, we have in this script which is visible, but you can not figure out what it is, right?

And it is actually on tabulates. So, it is when physically available, but you can not spend what it is.

(Refer Slide Time: 01:08)

Vedas

Transmitted across atleast 3500-5000 years without differing versions
Including exact pronunciation!
"UNESCO proclaimed the tradition of Vedic chant a Masterpiece of the Oral and Intangible Heritage of Humanity on November 7, 2003"
What "technology" used? Redundancy!
Various "pathas" of Samhita text: can recover from a corrupted text due to added redundancy: RAID-like! (Redundant Array of Indep Disks)
Pada-patha: each word in its separate form
Krama-patha: connects a word in pairs
ABCD becomes AB BC CD DE... ("2-mirroring"): 2 copies
Jata-patha: ABBAAB ("3-mirroring"): 3 copies of A, B, ...
Ghana-patha (ABBA ABCCBA ABC BCCB BCDDCB BCD...)(("10x")
Metrical (similar to checksums!) & Musical
"Information dispersal"
Human Reproduction! (Oral transmission)
Use efficient "virtualizers"!



Whereas in the case of whereas, has now a written down, but it is accessible now, it is written down for a long period time only in the recent last let us say thousand years. So,

it has been written down. So, it always transmitted orally, but a survived. So, there is some interesting thing about how it happened, and you were discussing that there is some lot of technologies were used to make it happen. One of them was a redundancy and I went through some of this, right?

(Refer Slide Time: 01:36)

FROM sanskrit.safire.com

पदपाठ 1,2,3

तम् । भागधेयेन । वि । मुञ्चति । प्रतिष्ठित्ये । यया । रज्ज्वा । उत्तमां । गाम् । अजेत् ।
 । ताम् । भ्रातृव्याय । प्र । द्विशुयात् । निर्मृतिम् । एव । अस्मै । प्र । द्विशोति
 ॥ तै सं २-२-६-५ ॥

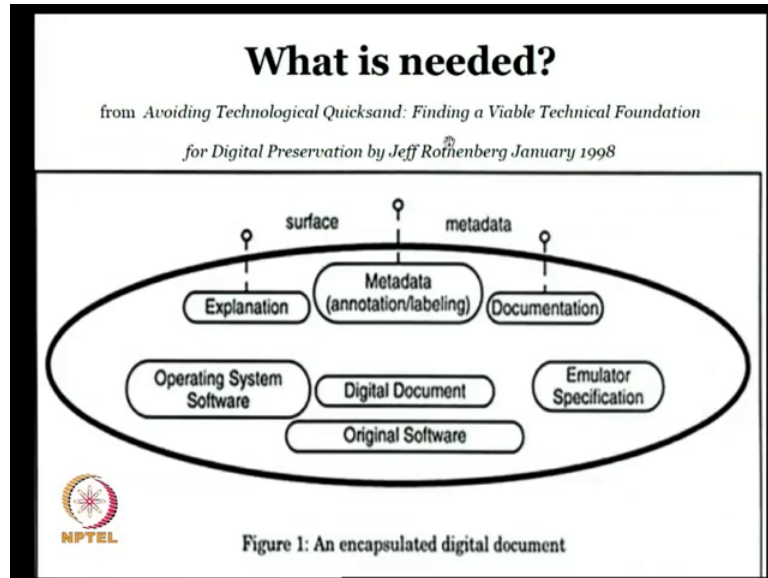
धनपाठ 1,2,2,1,1,2,3,3,2,1,1,2,3.

तं भागधेयेन भागधेयेन तं तं भागधेयेन वि वि भागधेयेन तं तं भागधेयेन वि ॥
 भागधेयेन वि वि भागधेयेन भागधेयेन वि मुञ्चति मुञ्चति वि भागधेयेन भागधेयेन वि
 मुञ्चति । भागधेयेनेति भागधेयेन ॥
 वि मुञ्चति मुञ्चति वि वि मुञ्चति प्रतिष्ठित्ये प्रतिष्ठित्ये मुञ्चति वि वि मुञ्चति प्रतिष्ठित्ये ॥
 मुञ्चति प्रतिष्ठित्ये प्रतिष्ठित्ये मुञ्चति मुञ्चति प्रतिष्ठित्ये यया यया प्रतिष्ठित्ये मुञ्चति मुञ्चति
 प्रतिष्ठित्ये यया ॥
 प्रतिष्ठित्ये यया यया प्रतिष्ठित्ये प्रतिष्ठित्ये यया रज्ज्वा रज्ज्वा यया प्रतिष्ठित्ये प्रतिष्ठित्ये
 यया रज्ज्वा । प्रतिष्ठित्या इति प्रतिष्ठित्ये ॥
 यया रज्ज्वा रज्ज्वा यया यया रज्ज्वोत्तमामुत्तमां रज्ज्वा यया यया रज्ज्वोत्तमाम् ॥
 रज्ज्वोत्तमामुत्तमां रज्ज्वा रज्ज्वोत्तमां गां गामुत्तमां रज्ज्वा रज्ज्वोत्तमां गाम् ॥
 उत्तमां गां गामुत्तमामुत्तमां गामाजेदाजेदामुत्तमामुत्तमां गामाजेत् । उत्तमामित्युत्तमाम् ॥
 गामाजेदाजेदां गामाजेतां तामाजेदां गामाजेताम् ॥
 अजेतां तामाजेदाजेतां भ्रातृव्याय भ्रातृव्याय तामाजेदाजेतां भ्रातृव्याय ।
 भ्रातृव्याय भ्रातृव्याय भ्रातृव्याय ॥
 तं भ्रातृव्याय भ्रातृव्याय तं तं भ्रातृव्याय प्र प्र भ्रातृव्याय तं तं भ्रातृव्याय प्र ॥

From sanskrit.safire.com/KYV2265.html

So, we looked at the some other ways that was done.

(Refer Slide Time: 01:41)



So, in general the problem is extremely hard. It is tremendously hard problem that is why it was there is no proper solution still to look at the kind of thing that is required you can

there is an interesting person by Jeff Rothenberg, he is talking about avoiding technological quicksand and finding a viable technical foundation, but digital preservation these are very important problems. Because a lot of stuff that we are doing nowadays is in the digital form now. If you have your photos on some gadget now, and nobody can find a way to read it 10 years from now; now you've lost a lot of time.

Now, people are putting land records in digital form. If 20 years from now or 30 years from now nobody is able to read, it is a disaster. The problem is that this particular technology or technology currently is in that situation. So, that is why what we are trying to say is what is it that we have to do. For example, you have a digital document that actually; for example, let us take some ancient software that you might have heard about something called WordPerfect 5.0 (Refer Time: 02:59) ran on a PC only, it never ran on a unique system for example, who knows PC will be extinct, and then the next decade the kind of PC that we were using is gone.

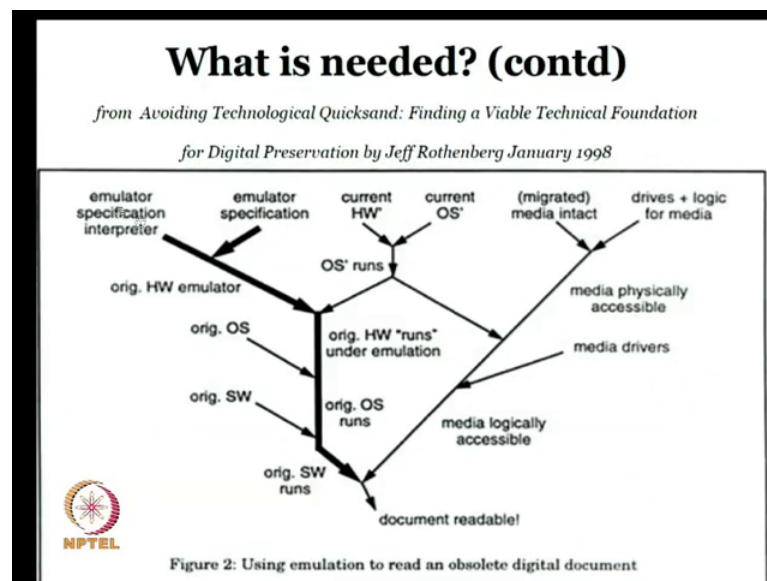
Now, for example, you have tablets, right. You know that, right. Now, the PC production has come down. And the tablet production is going up it has already crossed over there is more tablets produced than PCs. So, who knows the kind of PCs that we have seemed as a may not exist. So, the operating systems software may not run it might all be Android who knows in the future. And the original software WordPerfect may not be available. Nobody has taken that level to make it accessible 100 years from now. You probably might know that some whole languages are disappeared right. Natural languages and computer languages all disappeared.

So, there is also some documentation that goes with it, what the data is all about right, that it is population records or whatever it is there is some information that is kept. And there is some labels also. For example, you can say something like it is version one, it is encrypted, it is let us say they can be a lot of other interesting information about that that is a lot of OK. And so, these are all the things that go along with it. If you are talking about a document it is not a document by itself. There is an ecosystem of sorts, which actually helps you to understand what the document is.

So, when you want to transfer it 1000 years from now, you have to carry the ecosystem with you. If you are unable to do it then the document is basically a set of 0's and 1's nobody knows what it is just like the marks on the tablet you can see it,

but you can not make out anything from it. So, that is the situation that we are in, right. Now. So, our basic problem was how to make the digital document, server thousand years we can certainly make it. In the form of bits, you can transfer it, while I to do is to even take a CD; we tell a physicist. That guy will take the CD and look at the pits, he is able to say this is 0 and say one that does not mean anything. You can not figure out what it is just like the indus case just by the tabulates from the indus value civilization.

(Refer Slide Time: 05:26)



So, what actually we need the following? This is the complicated set of things that we need to do. First, we assume that the drives and logic for media somehow is accessible; that means, are what have done is, either I have a tabulate which I can see, or I have a physics kind of apparatus, right. Which can see the pits 0s and 1s, it can be in the reason of nanometers or some micrometer does not matter. In principle you can construct on apparatus which can see 0s and 1s.

So, basically this means that I am able to either use the drives and logic for media either that, or I am able to somehow migrated somehow thousand years from now, and able to see the 0s and 1s. I am a this point I able to see 0's and 1's a this point that does not mean anything actually, but this is the basic thing without this that of course, nothing can be done, but once you have 0's and 1's, you need other things. Because once you have 0s and 1s then you have to supply an operating system that looks at it, and says what to make of this. It may be that the 0s and 1s basically something like you took a CD-ROM

it has got bit some or a disk for example, it has got some sectors the initial sectors will tell you what is the master boot record partitions. All those things that also will come as 0's and 1's from the very beginning in the beginning of the device that only the os etcetera, I can understand because that knows what it is suppose to make that it knows that that is the master of boot record this is a partition information all these things.

But once you have the os information that is able to use that to basically allow you to figure out where the data actually starts from what block level it is. Is it at the level of let us say sector level? It at the level of 4 kilobyte blocks? All those kind information. In addition, nothing says that you are able to really directly of because it could be that it is encoded. It may be encoded for liability like we looked at the previous case where you wanted to preserve, the vedas basically encoded in somewhere, right. You have to d do you have to basically reverse their encoding process decode it, right.

To see what is all about because finally you are looking for the data similar it can be compressed you have a decompress it. It might be encrypted you have to de encrypted, all the stuff also is there. Now once you do all those things finally you have the data. But that is not enough because this data has to be understood by the program that makes sense of it. For example, if you take a word document. A word document has come from many versions, right. That is a word 6 words I do not know whatever 7 8; I do not know whatever right.

Now, there is something called I think there is a version for windows 7 I think, there is some there was there for windows 60, etcetera, right. And if you give the old version of the thing to the new one that the access, I can not read. It I can not understand it, because each of the programs have written something in that file in different ways, and it can not really unless the it is understood by a program it can not figure out there to start document. From what to do with the document want to take the bits from to figure out and how to show you to you right.

So, essentially you need to have the ability to run the application. Whether application actually runs on a particular operating system right; that means, you have to make the operating system run. And but if you want to run the operating system it may be depending on a particular hardware itself. It may be that it is only specific to x 86 and probably who knows x 86 will be absolute a 3012, it will be analyzing of course, if it is

alive in 3012, but let us say, but; that means, I need to be able to run the architecture also in some sense; that means, I need to have an emulator. An emulator requires an emulator specification, that emulator specification on the interpreter have to be there then only the finally, the hardware x 86 available, that x 86 stuff is required to run the operating system, right. And basically, you are taking you have a current of hardware and current operating system in 3012. And then using that I am able to run the original operating system on that original hardware. Then the original os is running, then the original software is running, and the original bits are available, then you get the document. We can say how complicate the whole thing is if not rival thing.


So, if you want to do this, it is a mind blowingly complicated thing. That is why nobody guarantees any of these things as of now people are thinking about how to do it a very interesting solutions, but the simplest of course, is anything that is valuable, you store it and keep on moving it to newer technologies. That is your business. You have to keep on moving it. When able to do it, you lose it somewhere. It is like the way we have lost languages is really lost scripts. We do not understand like where we lost indus valley civilization script. Because we do not know what it is now various in spite guesses, but nobody knows what it is still.

So, same thing is going to happen the digital media unless there is some special care they are all excited about digital media because it looks great, but there is a fatal weakness in digital media at the storage level.

(Refer Slide Time: 11:17)

Film Archival

- Data preservation requires active energy to move it from one format to the next new generation format
 - needs to be done every few years
 - cost is so high that many movies shot digitally stored in analog form as a fallback in case the migration unsuccessful
- Storing a digital master record of a movie costs about \$12,514 a year, versus \$1,059 to archive a conventional film master in a salt/limestone mine
 - US Academy of Motion Picture Arts & Sciences after a yearlong study of digital archiving in the movie business (2007)



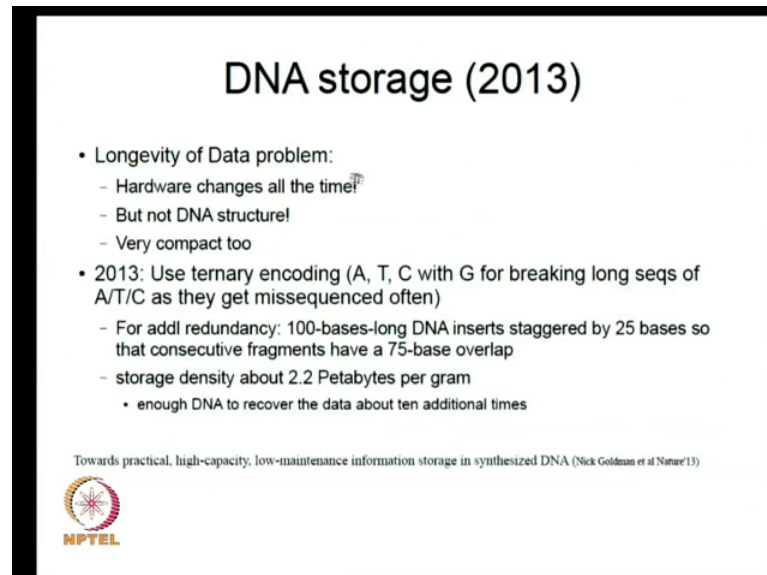
Again, one more example of this serious problem. In film archival it turns out if you want to preserve the data let us say you made a movie. Now you want to keep it around for 100 years. Data preservation requires active energy to move it from one format to the next generation format. I think all of you know about 35 mm 75 mm or it is 70 mm from this various formats are there, right. You have to do it every few years. Cost is so high that many movies shot digitally stored in analog form as a fallback in case of migrations unsuccessful. Why is are migration unsuccessful? Because you have to create all this infrastructure even in the case of media you want to do it in the case of films also.

If someone one person gets something slightly wrong, and that information is lost it; may be very difficult to get it back. It is similar to you have a document, you encrypted it, and by real bad luck somebody lost the key, you lost it that also is a problem. So, this is one aspect the other thing is storing a digital master record some people also done some studies some us academy of motion picture arts and sciences. They have looked at this problem, storing a digital master costs so much whereas, this much for is a factor of 12 difference by a condition study; to store a conventional film master in a salt limestone mine you keep it because it is cool, and it is a case sort of natural refrigerator naturally cool that is usually keep it in the salt limestone mines.

So, it is a factor of (Refer Time: 12:53) and the done a serious study of all these things, because you have to think about costing all the machines in an etcetera. Because finally,

if you want to preserve from one format there is some machinery some servers are needed etcetera. This is one instance of it.


(Refer Slide Time: 13:12)



DNA storage (2013)

- Longevity of Data problem:
 - Hardware changes all the time!
 - But not DNA structure!
 - Very compact too
- 2013: Use ternary encoding (A, T, C with G for breaking long seqs of A/T/C as they get missequenced often)
 - For addl redundancy: 100-bases-long DNA inserts staggered by 25 bases so that consecutive fragments have a 75-base overlap
 - storage density about 2.2 Petabytes per gram
 - enough DNA to recover the data about ten additional times

Towards practical, high-capacity, low-maintenance information storage in synthesized DNA (Nick Goldman et al Nature13)



So, let us look at some very interesting solution this problem. This is what is called DNA storage. I mention that in the case of you can actually use some apparatus to check the 0's and 1's. So, you can always get back the 0's and 1's.

So, some other persons have come with idea that why not store it in DNA itself. Because DNA's what is it is got an alphabet of 4 it is a 4-alphabet system right. So, if DNA survive in 3000 (Refer Time: 13:48) hopefully human beings are still around. So, DNA also will be the same approximately same.

So, idea is to use some encoding, and actually some people who have tried it. They have essentially, they get extremely high densities, something like 2-point petabytes per gram. Even if you give an excess amount of about 10 times excess grams of DNA's kept. So, that even if you lose one part of it you can get some. So, there are various people doing this kind of stuff, but here also you can see the fundamental problem is not solved. What is the fundamental problem? I need to capture all the associated metadata about the document that has to be carried forward. What is been eliminated here is only the hardware played things. For example, if you go for a DNA storage, what has been eliminated? You are able to get to this point. Which are too much. This part of it has not been handled, but this part is handled. You are able to see the bit patterns, very without

too much trouble you do not have to worry about all these things. And you do not have to do the migration. One great thing about DNA say DNA is DNA, you will be do not have to migrate it.

So, this part of the thing is already taken care of. So, it might be a good solution for this film archival guys possibly, but they also require programs to read their movies. I think you all know about varieties of movie formats, right. You know something what mpeg 4 there is something called mpeg 1, there is who knows what is going happened in the future, right. Where is things will be there, but that problem has not been solved. Because you have to store that information also in a form is accessible, some 3000, 1000 years from now.

So, what is this DNA storage? Basically, you want avoid hardware changes, right. And DNA does not change therefore, it is a good model. So, you can take care of the hardware problem. It is very compact to that is why you can get very high density. This much higher than what we can do with current semiconductor devices. It is really amazing it is 2 point a petabytes per gram. So, what this is a paper in rich nature recently just few weeks back, towards practical high capacity low maintenance information storage and synthesized DNA.


So, they use ternary encoding instead of quaternary encoding, because it turns out the some of these things there is a problem with the sequencing. So, they use one of this alphabet characters, right. G you know that in DNA there are 4 of them A T and C, right. And the G is the thing they use it as a way to break it up into smaller pieces. So, that the sequencing is done. And they also provide some redundancy. So, they basically have 100 bases long DNA was staggered by 25 bases. So, that the consecutive fragments have a 75-base overlap. These very similar to this model. What we looked at this way, identical thing almost is said that this is none terms of basis here there doing it in the terms of words.

So, there also basically looking at some way overlapping DNA fragments. So, that if one DNA fragment gets into problem there is another DNA fragment which achieve overlap (Refer Time: 17:10) you can essentially recover the information.

(Refer Slide Time: 17:17)

Insurable Storage

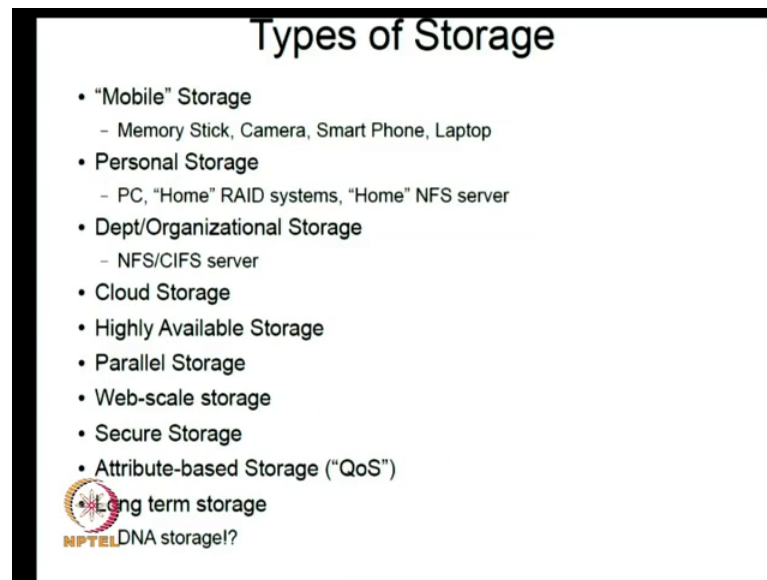
- Digital Documents as insurable property
 - Provide economic incentives for storage service producers and consumers to jointly create a marketplace for a diversity of differentially-priced services
- Insurable Storage Services



So, the other models also actually be propose this solution insurable storage. Basically, you make digital document has insurable property. Provide economic incentives for storage service producers and produce and consumers to jointly create a marketplace for a diversity of differentially priced services. Basically, you give it to google or facebook to the have a document you store it. Different people will they have to have some mixers by which they can migrate that things, and then I get a service from them. They will guarantee that my data is intact 100 years from now or whatever it is. If it is a document for example, like my land records I will ask people or your multiplicity of insurers saying that just like people insure homes, right. And I would not to insure digital documents, I tell then you charge in the so much I will make you please make sure that available 50 years from now. Because it is a document about a home which I want to use it as a legal thing. Those can this this things are not possible.

So, probably you will see all this stuff happening in the new future. So, one thing that you have to think about is that this storage it looks like a on the surface it looks very simple, but if it is look at it there are varieties of storage right.

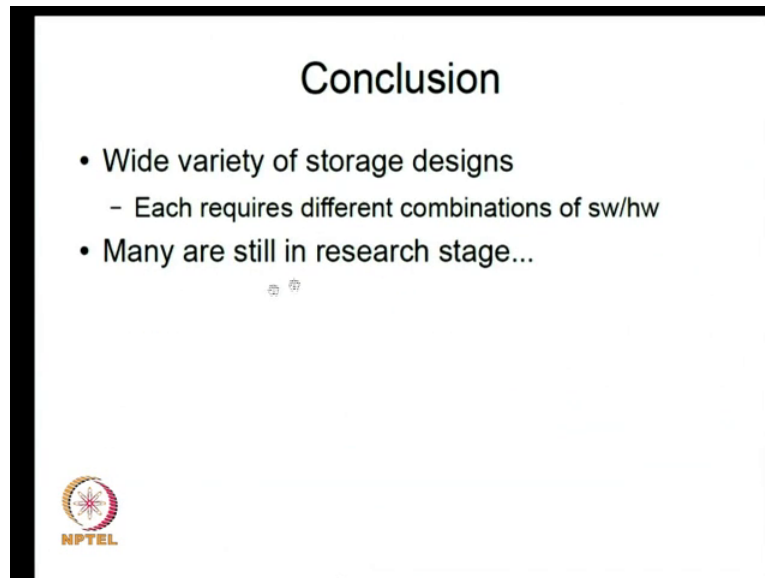
(Refer Slide Time: 18:33)



Each of them has its own needs and what it tries to do. And so, put a look at looked at some of these things, right. Some of them look a bit far out like DNA storage, but some of these things are going to happen, already has happened and the one thing which has not really seriously happened is attribute based storage. It looks very promising, but someone does not taken off because it is very difficult to guarantee, but best of them have been tried in some form other.

So, the last 2 are the ones which are not yet there in conventional medium, but I think I had told you, right. When it comes to preserving something for 5000 years, they have been some examples good examples where it was happened right. So, even probably start seeing some of those things in the future.


(Refer Slide Time: 19:25).



Conclusion

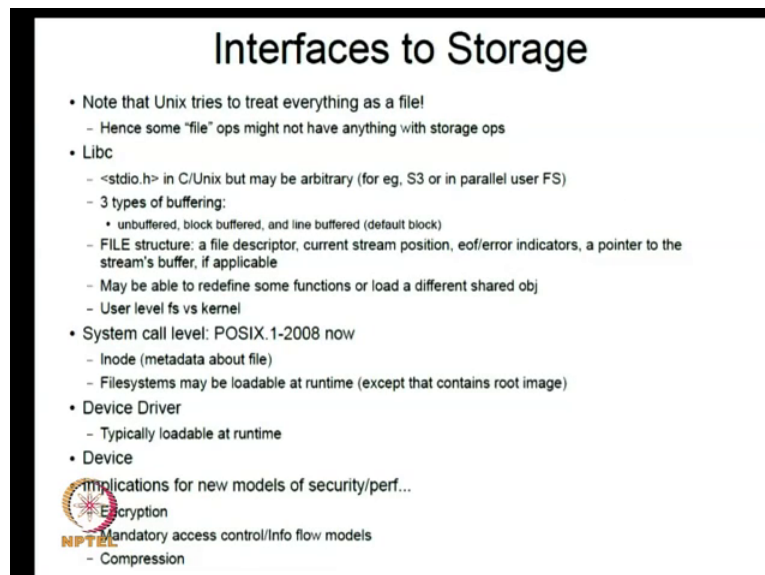
- Wide variety of storage designs
 - Each requires different combinations of sw/hw
- Many are still in research stage...

⊞ ⊞




So, quite a few of these things are in research stage. The basic problem with all the storage is that each requires a different combination of software and hardware. Different hardware and software we have to somehow gap on this problem. It is a very hard problem. I just want to make sure you think about it in that way.

(Refer Slide Time: 19:47)



Interfaces to Storage

- Note that Unix tries to treat everything as a file!
 - Hence some "file" ops might not have anything with storage ops
- Libc
 - <stdio.h> in C/Unix but may be arbitrary (for eg, S3 or in parallel user FS)
 - 3 types of buffering:
 - unbuffered, block buffered, and line buffered (default block)
 - FILE structure: a file descriptor, current stream position, eof/error indicators, a pointer to the stream's buffer, if applicable
 - May be able to redefine some functions or load a different shared obj
 - User level fs vs kernel
- System call level: POSIX.1-2008 now
 - Inode (metadata about file)
 - Filesystems may be loadable at runtime (except that contains root image)
- Device Driver
 - Typically loadable at runtime
- Device
 - Implications for new models of security/perf...
 - Encryption
 - Mandatory access control/info flow models
 - Compression



So, let us try to think about what we have right now. So, let us keep also some Unix model. Now the varieties storage possible and it depends on the system you use. So, if I suppose I take Unix as the model. One thing about Unix you have to remember is that

Unix tries to treat everything as a file. So, there will be a lot of code operations which seem like you are dealing with files, but there are actually dealing with something else there actually not actually doing anything with storage operations. A good example is; suppose you have the proc file system, in linux and other file systems power systems we have something called slash proc. What is slash proc system doing? It is exposing certain information about system in some easy form in a hierarchical manner. It could be for example, even things like cpu information, how many cpus are there on the system, right? It could be about how much memory is there is nothing do with actually storage for our practical purposes. What we what we mean by storage in this class, right?

But it is available to you as a file in a slash proc system, in the in the file system. So, it turns out if you look at Unix interfaces, there are lot operations which use seeming the files, but actually they are not released storage operations. That is something you have to keep in mind. So, let us look at what are the possible interfaces in Unix kind of systems. First of all, there is a library interface. You have the system call interface. You might have a device driver interface. You might have a device interface, as it listed a few of them you can think of a few more.

So, for example in libc, what is libc? It is a library that is used when you program in c library. C there is a corresponding one for c plus plus. There is something called I/O stream library file, right that is for. So, this standard I/O dot h is used in c slash Unix, but in c plus plus you might use something called I/O stream etcetera. So, the libraries that are needed for doing storage operations can be observe it arbitrary. But in c and Unix it is typically standard I/O dot h. For example, if you are using amazon s 3 kind of file system s 3 kind of storage service their libraries going to be different from any of these things. Again, if you look at this libc there is a system called, you will see that different things happen at different levels. In libc for example, the notion of buffering is very strong. Because application wants to buffer things what is application want to buffer things? Because application knows that storage devices are slow. And it is telling is library c, c library please do something for me to make sure that I am not hurt by the slowness of the devices.

So, applications typically want some buffering so that the performance of the application is reasonable. So, that the application isn't have to worry about buffering itself the underlying library slash operating system together are doing something for me. So, for

example, there are 3 types of buffering you might have come across unbuffered means do not any buffering for me. For example, there are things what are called character devices, right. Which do not need any buffering. You want to be given it access directly. We can having block buffered therefore, what disk devices CD-ROM devices dvds those kind of things, or line buffered terminals for example. You only when the line is complete give it to me, otherwise do not give it to me.

So, the basic reason is that the devices are slow. So, you want to amortize the cost of processing or the cost of waiting for these things so that if you get a big chunk 4 kilobytes. It is better than getting 10 bytes every time. It was going to discuss so slow. It is not worth it. Because we could say give me in 4 kilobyte chunks, and I will this libc will actually manage. It for me it will keep track of where I am, right. Now in that buffer. That is why there is a structure called the file structure, some of you might have used it extensively. And this file structure is a thing that is kept by the libc so that it can actually replace normally what you would have a device view with a buffer view. You think in terms of buffers we want thing in terms of devices.

Because somebody behind the back is actually managing that movement of information from the device to the buffer. You would think in terms of 5 structures you do not think in terms of device structures. For example, it has a file descriptor, the current stream position. So, you are in a buffer and you are processing it has some it has come sometime in the past from the disc or wherever to the memory, and the libc is keeping track of where you are in the buffer. It can also keep track of in the file or error indicators, sometimes when you try to read something there can be error.

So, your libc is keeping track of these things. Instead of the device keeping track of it. Normally what happens is that you try to read something the device is unsuccessful not able to read it. So, it keeps track of what the error is, and somebody has to pole the device saying what happened I asked you to do something you didn't do it please tell me what was the problem. That is the device might keep here it is being kept by your file structure. Essentially in some sense you do not want to deal with that device structures, things because you have multiplicity of devices. Each device might do it in different ways.

So, you do not want to go and figure out what particular device you are dealing with and do it. You want to do it in general generically. So, this libc actually gives you a file structure give the form is. So, one thing about libc and this kind of structure the application level basically, these are basically what is happening you are doing it more as an application level you are in the same address space. So, you can actually replace your libc. You can change your you can have a alternate module, which interfaces for which gives you the interface for storage. You can load a different shared object, or you can redefine some other functions etcetera, that is up to you. Because it is under your control. It is under user space you see a address space, you are welcome to what you want to do with it.

So, it turns out that if you are using a user lower file system, then you actually augment instead of using the libc functionality the standard libc functionality you might augment it is your own functionality, if this user level file system. What are mean by what do you mean by user level file system? It means that you have defined your own file system for example, it is not what is supplied by the kernel. And this often happens in parallel file systems. Because parallel file systems they find the kernel file systems are optimized for some other design point.

So, the parallel file systems do not find the kernel file systems very efficient in some cases. So, you want to do your own? But we find then the kernel file systems are good for managing that device as a whole. So, you write something on top of it. And you directly use a user file system to deal with the devices. So, the operating system is still dealing with the devices, but you are not using a kernel file system, you are actually by passing the kernel file systems and defining user level file system, and that is directly dealing with the devices. You might still use device drivers supplied by the operating system. So, you are still going to device drivers.

So, there is some kind of a kernel facility by which you can access these devices, and the standard way again is using the file descriptors. Basically, open the device you open a file you get a file descriptor, you open a device you get a file descriptor. Because that is a trick that Unix did. So now, we have access to the device as long as permissions met to the device now you can manipulate they where you want, and this particular parallel file system user level parallel file system is going to talk to devices directly same thing is done by data. Basis also data basis might not want to be on top of a file system.

So, they might want to directly access the devices. Again, they will open the devices and you get a file descriptor and start manipulating it. Directly it may be a block device whatever it is, but the thing is that it is up to the application on to use it they have the permissions to use the device they can go and do it. Now that is one type of interface you can also have what is called system level interface.

Now, this there are varieties of interfaces that available it has been changing for some time, the most common one in Unix world is what is called Posix. It has started something called Posix 1, then there is some added they added some things to is called Posix b something added one more thing Posix c. Then wherever part is IEEE another people got together now there is a standard called Posix.1 2008.

So, most operating systems in Unix which are commercially sold they usually support one of these Posixs. It could be Posix 1, Posix 1 b whatever. Then may not support all the versions, but Posix.1 2008 is the most current one if you look at linux linux is mostly there it does not support it completely. Pre bsd also same story, all the 3 operating systems do not support this completely typically this is supported by commercial operating systems. It could be ax, it could be Unix where etcetera, those kind of people usually they support this standards. You might call industry standards. Here you can see in the libc case in the standard c Unix can we had a file structure, that is a metadata board the data you are talking about. In the case of Posix usually the metadata that you keep is the kept by a kernel is called the inode.

And here the typically there is a file system which is giving you this interface, and it may be loadable at runtime. A file system itself can be loadable runtime the only things that cannot be loadable runtime. Or those that contain the root image example in a booting the system you have to read the kernel, right. And the kernel is going to be sitting from there. Actually, has to sit in the file system. If you look at Unix for example, it is something called slash boot. So, it has to reside in some file system, but that cannot be loadable at runtime, it has to be part of the kernel image itself. Or it has to be part of something which is there in the bios itself. So, the it can go an it is able to interpret, the particular partition as containing a root image therefore, it can say go and read this particular root image, and using this booting process. It might you know multiple steps multiple stages it can bring that whole thing, but the thing is that now the bios should have the capability of locating on the disk for example, which partition has the current

bootable current booting system, operating system. Corresponding that it has to be able to locate the image that will be used as the kernel right.

So, it has to have some ability primitive ability to be able to locate this things, then was going to read that, and then read from that might call it the file system, that basic file system the slash boot file system in the case of linux. I am able to do that multistage booting process by which the kernel finally, gets into memory and then finally, it starts. So, typically many file systems can be loadable at runtime, but the ones it contained by you have to be handled separately, specially you have to handled.

So, this is one type of interface, there is another type of interface at the device driver, level we will come to that later basically this also a typical loadable at runtime. This is very specific to operating systems. People have tried what it called uniform device drivers, extensible device drivers basically the stuff operates across operating systems. But that has not been very successful, but people are there are some solutions in this area also we will look at this sometime later. There also interface available the device level for example, a device could be a sorted device or a SCSI device SCSI device it might speak what is called a SCSI protocol that is in interface there.

Now, if you look at all these things, you will see the many many interfaces, and every time you come up with some new models. Then this may have to be changed. For example, I might have new models of security or for attribute based storage, right. I might have some models for performance also, right. For example, I might able to say in some of those let us say in the future who knows, there might be an interface it says please give me this piece of data, but if you are taking more than x amount of seconds do not bother. Am I interested on that is a way it is done in the case of locking for example right. You may told that in the case of semaphores, there are something called real time semaphores by which you can give it some delay. You can tell the operating system I need a I need a unlock or a semaphore, but I have real time deadlines. If you are not able to do it, then I am not interested, something equivalent right.

So, there could be probably sometime in the future there could be such interfaces. Same thing what encryption; you might imagine that there are solutions which say that this is a file encrypted please decrypted with this key and give it back. That could be one interface usually that is not done. Because there is a slight different model even if use

encryption what really happens is; you have as some administrative procedure by which that particular file system is available and encrypted form, let me explain further.

Now, you can encrypt at the file level, you can encrypt with the directory level, or you can encrypt what the file system level a multiple possibilities. Now if you are looking at the support of an encrypted system and the file system level. Typically, what they do is they give you at the file system level. Now in file systems have the notion of mounting. What is mounting? As we have discussed in the previously, mounting is a way in which there is some tree of information, which is grafted on to slash root or some file system that is already existing, that is already visible. That is already visible to the program. You are basically taking some tree, it moved could be there on a CD-ROM, it could be there on a usb stick whatever, right. There is a file system structure on it, that you want to graft on to the main files the main file system hierarchy, that is already available to you right.

Now, what you can do is at the time this mounting is going on. You can demand that you I that you decrypted or you uncompress it. Or you install some specific thing, by which there is now some kernel related model which will intercept any access of an encrypted file or a compressed file. It is you normal access it as you as normal you just do it as what you normally do right. But norm because of this mounting operation it has added some additional module typically in the kernel.

So, that now anytime you access the normal way, because this mounting operation has said that this particular thing has been these encrypted thing or whatever, right. I have (Refer Time: 37:07) module it will intercept every single call of yours, every single access of yours. And it interpose itself, and then do the decryption on the fly for you or decompress it on the fly for you. So, that is even why even if I encryption compression, you do not find that these things are actually changing. It is not as if you are going to say read a file you will say read this file and say this encrypted please decompress this key. I do not give you the argument that way. You are doing it at the file system level at the mount time you actually make all this magic happen. So, that it is you do not have to say it.

So that means, that your application will work independently whether it is compressed or decompressed. Whether it is encrypted or not as long as this file system management operation of mounting has been done properly. The other reasons other things also you

can do, there are what are called you might have different modes of security, these are some traditional models. For example, you have something called mandatory access control or what is called discretionary access control. What is that standard one what do have is what is called discretionary access control? What are that mean? It means that if you give me some piece of information you are given it. So, it is mine now. I can give to anybody, nobody can the party will give to me. It does not have any control about where it is going to go. Whereas, in mandatory access control even a somebody gives it to me, the party you gave to me still controls whether I can give to somebody.

So, why this important? It is important because suppose I give you some sensitive information. I am giving it a because you have go it an office where you have to show me show them some piece of information. I have, but I am still worrying about what happens to that information. I want to make sure that it does not go into the wrong hands, right. If I am worried about where that information is going? Then have to use mandatory access going to. If I am not concerned about it, then I can use discussion access control.

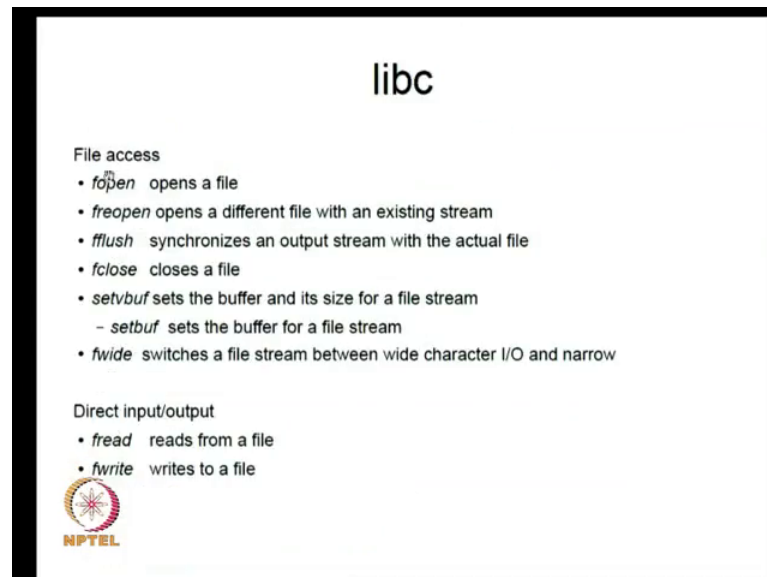
So, typically this is because the these models are also appropriate for file systems. Because they deal with information. So, it might have discretionary access control models or mandatory access control models. So, this have to be enforced by the file system, or might be who knows. It might be depending on at bit level a working in all these (Refer Time: 39:54) worried about; that means, that when you do a read of a file, you have to figure out what is it is whether it is allowed for me to give it to somebody else.

So, the question for us is where is information. Is it something you give it as a part of the arguments? Or is it that you have some information that is available the file level, but it is not given when I am being a read on write. So, normally the way you do it is by there are some special interfaces. Typically called I/O ctl interfaces I/O control, or FCN or file control FC and TL. There are some interfaces available specially by which you can tell the operating system or the file system that this guy I wanted to be tagged with a particular level or who can see it etcetera. There is some I am going to provide some information. So, what this mandatory access control models can be implemented.

So, normally the usual thing is you do not modify these things. Just like I mentioned about if you have encryption and compression you do that runtime, right. For mandatory

access control information flow models probably do that. You specially say that this a file; I created this file, I tag it to special properties. That this is this has to be managed mandatory access control. So, so it is not given as an argument whenever I do a reason right, but you do it specially using for special interfaces.

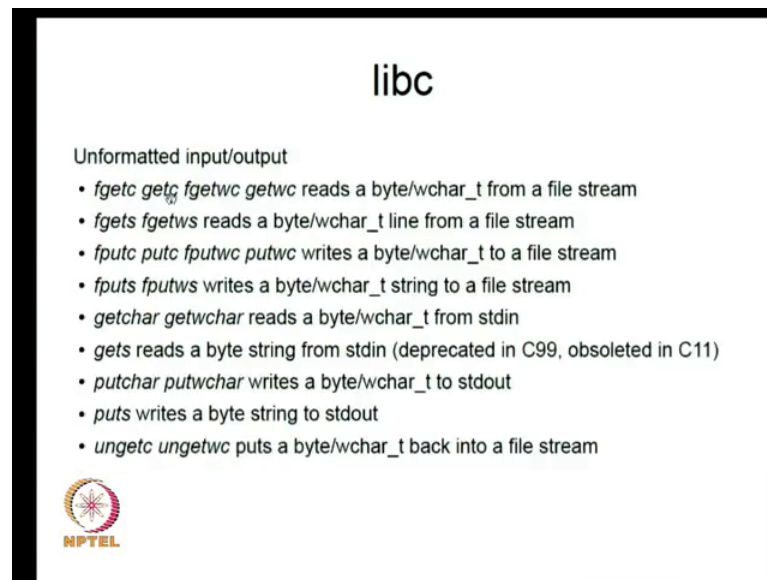
(Refer Slide Time: 41:55)



So, when thus quickly take a look at some of the interfaces. I think may have some of this very familiar to libc for example, opens a file, but it opens a file under some buffer policy. As I mention before the buffer policy is typically block this a typical thing. So, you have fflush, I am not going to go through all of them I am just to going to give a few of them. Fflush basically because it is buffered, it may be in memory I want to make sure it is going to be synchronized with the actual file that is fflush, I am done with the file fclose.

So, these are stuff which basically setvbuf basically sets the buffer and it is size. So, basically what is the sets the buffer means whether it is line buffered; whether it is block buffered or unbuffered, that is what this is and it is size also. So, there are other things like a fwide (Refer Time: 42:57) I am not going to go into there is also direct input output fread and fwrite. This is a part of libc.

(Refer Slide Time: 43:05)



You can also have lots of other types of operations which operate on a file stream. You can a *fgetc* said because a now there is the motion of a white character in the white character. While the white character come in we had something called unic code, the 16 bit codes.

So, because of that there are something called byte characters. So, typical byte code can only handle ASCII and a few character sets. Whereas, you want to encode many languages in the world, right. It turns out the byte ASCII code is not sufficient. So, you have to put a white character in 16 bit, but it turns out the 16-bit code also is not really properly well thought out. So, that is why there is something called utf 8 it is not a 16-bit code it is actually a variable length code, it can handle both byte as well as 3 bytes for example, like for example, if you are talking about indian language encoding we can we typically use 3-byte codes. And the good thing about this utf 8 is that you can mix ASCII and another indian language for example.

So, but you notice that interface here does not talk about 3 bytes codes, like something somebody, you have due to in application it is not part of, but whereas, if I using one type of unicode 16 bit, then this white characters available to you. So, you will see that there are things which read a byte from a file stream. You can write to it. You can get it from standard in standard input you will see that something will gets which has now

obsolete, because there is what is the problem gets, the problem is that it is insecure. There are what is called buffer overflow attacks possible.

So, it is now in c 99 means c 1999, they are basically deprecated it. Saying that do not use it, but they didn't bar anybody from using it. When the deprecated, it in c 11 2011, they are obsoleted in; that means, if you are standard compliant c, you do not you do not at support gets. You can throw it out and nobody can complain about it 3.


So, there are things which can write put character puts and get put it back into a file stream. There are these are the kind of stuff which is therefore, and formatted.

(Refer Slide Time: 45:59)

libc

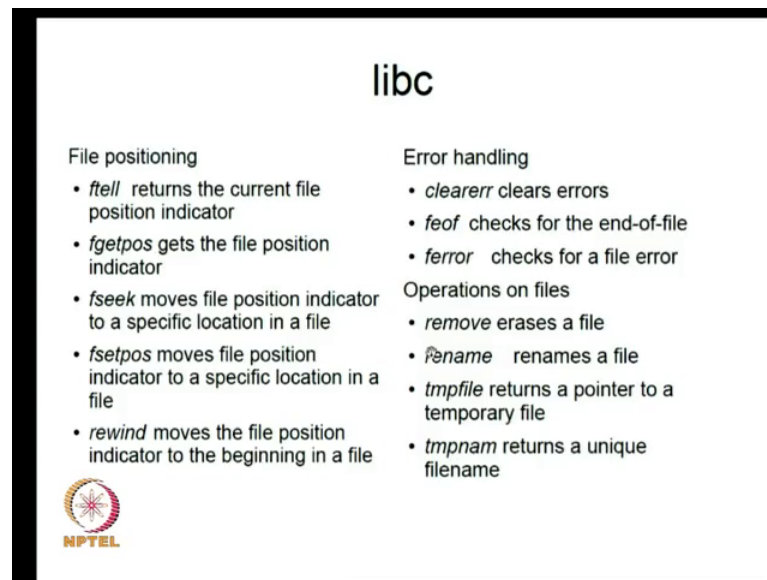
Formatted input/output

- *scanf fscanf sscanf wscanf fwscanf swscanf* reads formatted byte/wchar_t input from stdin, a file stream or a buffer
- *vscanf vscanf vsscanf vwscanf vfwscanf vswscanf* reads formatted input byte/wchar_t from stdin, a file stream or a buffer using variable argument list
- *printf fprintf sprintf snprintf wprintf fwprintf swprintf* prints formatted byte/wchar_t output to stdout, a file stream or a buffer
- *vprintf vprintf vsprintf vsnprintf vwprintf vfwprintf vswprintf* prints formatted byte/wchar_t output to stdout, a file stream, or a buffer using variable argument list
- *perror* writes a description of the current error to stderr



You can also have formatted, I am not going to about too much into it. So, basically you can do it to from standard in file stream, or you can have a string buffer, you can do it a string buffer. So, you can also have with variable argument list. You can print it also to a you can also print variable argument. And also, there is something which tells you about what is the current error, perror.

(Refer Slide Time: 46:36)



So, you can see this is all part of libc, they also other things like file positioning error handling and some operations and files. File positioning returns the current file position indicator I want to know where I am in the buffer or there exactly it is. Fseek we can move it a particular place. Rewind you can go in the beginning. Again, this depends on the device for example, a rewind might mean for a tape device to go in the beginning, right. Which is a physically, let us say done what it can also be done virtually, because nowadays what is happening is that tapes may now be emulated through a disc.

So, when you do a rewind; that is, being done on a emulated tape. That means, finally, what happens is that on the disc you actually go to particular sector somebody's going to do that for you all these things are basically multiple layers emulation could be going on here. Error handling clear error clears error what do you need it? Sometimes there is a device error and just like for example, on a printer sometimes there is a printer jam. You finally, after *clearerr* jam you say that there is no error please proceed right.

Similarly, here also in devices tapes and other kinds of you may want to clear the error, once you have taken care of it. So, the things like checking for end of file etcetera. Check whether I do the operation has it result in an error for example. All this are available libc again this is since it is libc you do not know what device you are talking about. Because as long as it comes as some file. A file can be slash war slash something, slash home

something or it can even be slash del slash something. Actually, could be device it could be a tape it could be a CD-ROM it could be anything.

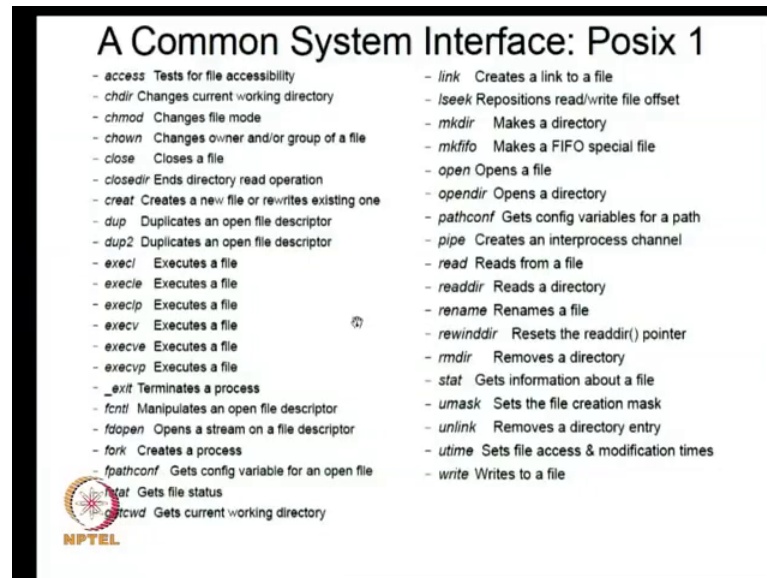
So, this libc handles all this at generality in complete generality, it can also be operational files. You can there is a file you can rename a file, these a very complicated operation because you are you could in principle, rename, directories also. You might actually take it from one part of the file system hierarchy and template somewhere also. And this can be very complicated because in a concurrent system, there are lots of people who could be sitting on each of those file system hierarchies.

So, to ensure that there is no activity here before I can move there. It might it requires some care it is not trivial. So, this is a usually a extremely complex operation, but a very important point because all of us depend on this. For example, if you are doing some editing what you want to do is; you want to keep the old file, you read the whole version of what you are reading, it should come to your buffer, but one we say, right. Right I want an atomic operation by which the current version as a as seen in the editor, right. Should atomically replace what is there in the disk, we should atomic. If you do not do it atomically then you may lose your file.

So, this a very critical operation, I think those have people who have used vi for example. In the early 1970s it was not atomic. You write it you may or may not get your file. You may the file may just disappear. So, it was a very scary thing in now was days luckily now all editors typically is the rename operation, and it is guaranteed by the kernel. That either you have the old thing, or the new thing will not have some in between stuff or typically will not was anything typically they guarantee that you are safe.

So now that is basically at the libc at the libc level right.

(Refer Slide Time: 50:33)



You will have it at the posix level also. This a fairly long one. So, may not have time to go through much of it, but there are various things for example, you have something called access, tests for file accessibility and so on. So, I have this permissions can access this file. Now again this is a in a discretionary access control can a mechanism this might be fine. But in mandatory access control are what is called information flow models, right. Trying to even know whether I can access something also is information link for example, if I in some cases, right. The information where I can access something itself is information.

So, in in this kind of information flow based systems, this access maybe severely the functionality will may be seriously limited. They might only say that among your own class of users, you may be able to do it not across other classes of users, right. I think the other things you might have come across change directory, change the file mode, change the owner shape close a file, you can also use directories you can open directories for example, there is this open directory there is a corresponding closed directory, you want to use it as a you want to go through the each are the entry, in the directory for that you can open a directory and go through one open directory read directory, and close directory read directory means you want to go to the next entering the system.

So, again this is there. So, that who knows the directory itself is it is a huge directory, and it is hashed example let us say that I have as a mentioned to you. My main directory

has got 18000 entries. So, I am searching for something somebody might had a hash. So, asking it to go from one step to the next step, right. I can not linearly read it. I can not read it from one next is not like that, because somebody is doing hashing for me. So, I will actually go through a loop I say get the next one. And system has to figure out going through hashing other things which is the next one. It is keeping track of it some in it is own I do not know how it is done.

It is transparent to me, but I have the capability of saying open the directory, go to the first one, give me the next one give me the next one. How it is done inside? I do not care, because it is not using hashing it might use compression I do not know, I do not know what it is doing, I couldn't careless. There other functions like create and it has got very fairly complex semantics, because of these kind of complexities if you are going into new domains like cloud storage or high performance parallel storage, people try to eliminate this kind of complicated semantics, usually they do not support this kind of function.

For example, we know that there is something called opening a file, right. There is some medium that is common in UNIX systems, where you open a file and delete it. The idea being that as long as it is open, you have access to reading and writing that particular file, but the minute you close it or the system crashes that thing disappears. In some sense it is like a temporary file in memory sitting. There you open it and then you delete it; that means, that whatever you put in is available only before that session process, right. A process, right only for that period nothing more nothing less.

So, various idioms so that kind of there in Posix. But these idioms are very difficult to support and other configurations, in other systems. Like, I mentioned cloud storage and all these things are very difficult to do it. So, these things these are the first things to be thrown out. So, again this is a complete set of Posix, once you will see that there are some initial ones, for example, pathconf. What is it? It turns out pathconf what is it basically you might have a device. Slash del slash are empty something. You want to get a configuration variables path basically that is a device actually, or to get the; what is the configurations of that particular tape device, or the CD-ROM whatever it is.

While it could also be for example, let us say that you have a file which has what is called an extent base system. What is an extent base system? It is a system which tries to

allocate not blocks of 4 kilobytes, it can try to allocate in very long contiguous chunks. It can be 16 megabytes or 4 megabytes. Because we know that discs are very good at giving you big chunks of information as long as they are contiguous. So, whereas if you do it 4 kilobyte at a time, this device performance is very bad.

So, you want to be able to say this particular file is told as an extent means a contiguous set of 4 megabyte set of blocks, and it is a parameter of the file. So, I want to be able to say what is that variable, because I have a file slash a slash b slash c, I want to get extent information. So, that when I write my application I know that configuration of this saying it is a 16-megabyte extent, I can use it to optimize the way I want to do it. I know the fact that 16-megabyte chunk is there. So, will so that there are we will see that there are a lot of other operations which are more kind of file. For example, exec actually uses file quite a bit because it uses something called memory map.

So, when you say execute a file actually turns out to be using a file system primitives for mapping. What is happening when you do in exec? There is a code that is there, right. And what you are doing is you are mapping the code segment on a particular location memory, and you are also taking care of the stack the data section etcetera and mapping it right. So, all these things actually go through the file system. Such citizen why all these things are included and unknown I am listing only the file system operations.

So, you also have functions like get files status. For example, you know want to know whether that file is locked or not. Or it is being currently let us say is it being updated, right. Now there is some types of information that you can get about a file and that is often used as a way to in case some heavy operation is going on; you may not want to wait for it. You get the status, and then decide what to do want to.

So, I will again continue from here next time. And then we will look at other interfaces in the system.