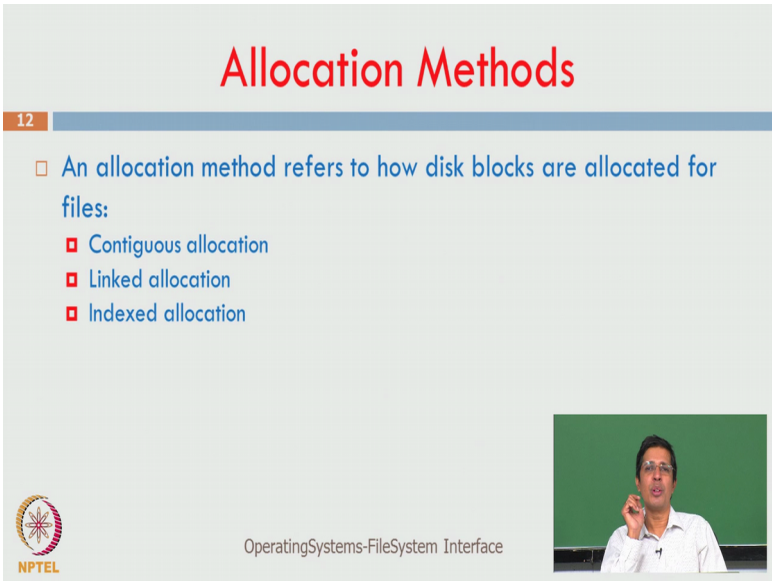
Information Security-3
Prof. V Kamakoti
Department of Computer science and Engineering
Indian Institute of Technology Madras
Basics of Unix and Network Administration
Operating Systems
Mod02 Lecture 12
Module 12: File Systems-2

So welcome to this module on a file systems 2, we covered some basics of file system in the previous module with lot of inputs that we try to give from a security perspective; we will see more about that in this module.

(Refer Slide Time: 0:26)

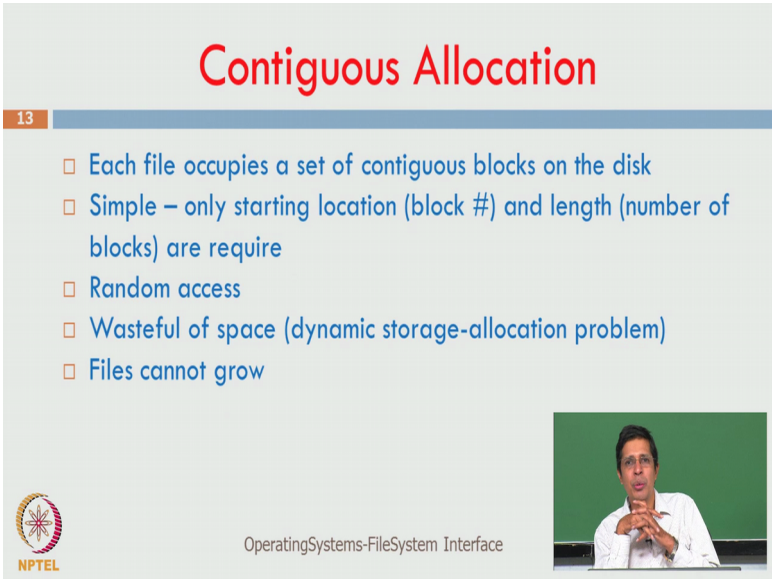


Now one of the important things is that when we looked at a file control block, there are lot of details about the file, its name, when was it created, when they what are the permissions etcetera. The next thing was, how the file is allocated and that is very very important, because a virus can basically go and look at this o allocation and if it can go and pick (())(0:50) this allocation then you may land up with lot of problems. So form information security perspective, how the file is organized, so I have say file with say 2 megabyte large file, how is it actually stored in the disc, right and that is what would be the subject matter here and understanding of that would be also very important for you to go and look at how some viruses can be basically go and (())(1:17) with your file system, right.

So there are three ways of allocating, so I have 2 MB how will I store it in the disc where will I store it in the disc? So when I am creating the file the operating system will allocate space. So there are three ways of allocation, one is contiguous allocation, another is linked

allocation, third is indexed allocation we will see all these three, the pros and cons of this, okay again these are basic operating system concepts, why I am revising it here is, because these would have very serious security implications, understanding of this would be useful and important for you to understand certain viruses.

(Refer Slide Time: 1:52)

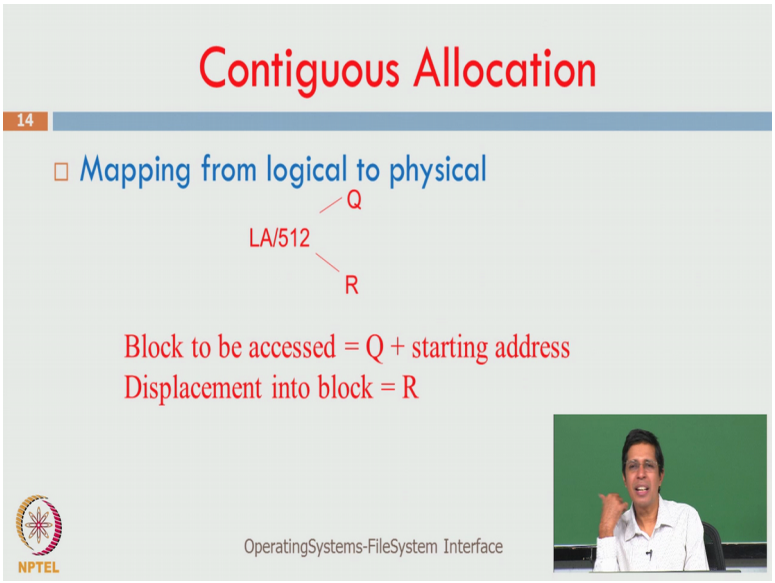


Now what is contiguous allocation? So disc basically consist of blocks, each block let us say can handle up to 512 bytes, sometimes even 1024, 2048 etcetera. Now suppose I have a file of say 100 blocks, these 100 blocks should be one after another, right. So that is very simple. So 100 blocks should be just behind one after another, so this is called contiguous allocation. Now what are the drawbacks of this contiguous allocation? the drawback of this is that the later I want to increase that file, today you have a file 4 MB say 2 MB or 4MB, tomorrow I am going to add some more records to this, you have keep editing the file, the file cannot grow, because you make one contiguous allocation of 4 MB then immediately the next file will start.

So you need contiguous allocation, so that is why if I am now go and change the file then the operating system has to completely remove this 2 MB and find another space which will have 3 MB and now copy this there. So this becomes extremely complex for the operating system to maintain. So contiguous allocation has this problem, but the good thing is that if part of the file gets erased due to some virus etcetera, the remaining part will be intact, so I can boot some dummy and then continue I will have this. So that is a bigger advantage of having this contiguous allocation.

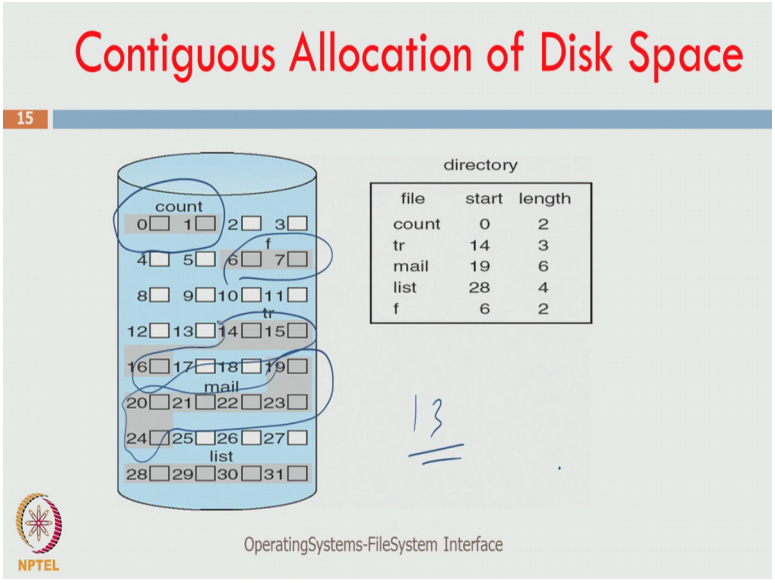
The third important thing is that if I want to allocate, so if you look at C or C plus plus I could go and access one particular byte random access file. So I can go and say I can access byte number 100 byte number thing, I can go peak, okay. Now if I want to do such a thing contiguous allocation is very easy for me. So if I want to have, so each block has 512 bytes if I want to access the 514 th byte then I can directly go to the second block and look at the second byte there, if each block as 512 bytes. So random access becomes very easy. So there are the pros and cons of contiguous allocation.

(Refer Slide Time: 3:53)



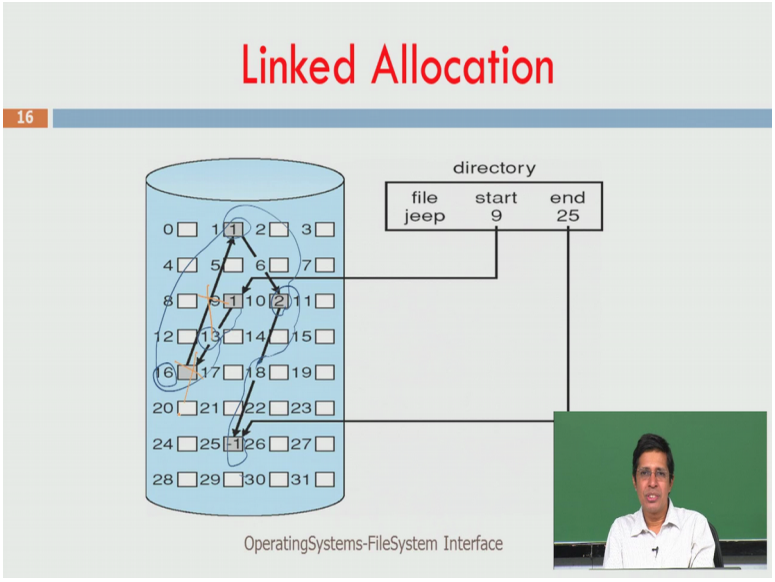
So if I want to access the some byte there, so the block to be address would be Q plus Q th block and R th entry within that block. So I can just Q plus starting address, so that will be the block within that block I need to adjust the R th byte. So I have a file of say n blocks say if I want to access the K th weight (())(4:13) I can generate which block that K th byte will belong to that will be k by 512 and K reminder 512 will give the displacement within that block. So I can generate this Q and R and then quickly access. This is very very simple.

(Refer Slide Time: 4:30)



This shows this explains you why contiguous allocation can become issue, right. Now, you see there is one file which is occupying 0 and 1, another file which is occupying 6, 7, another file which is occupying 14, 15 and 16, 19, 20 this whole bit. Now if you see, there are 1, 2,3,4,5,6,7,8,9,10,11,12,13 blocks that are free if I want to have a file which is asking for 10 blocks, I cannot allocate it now, I need to go and you know, so there is already some fragmentation 2, 3, 4, 5, 8, 9, 10, 11, 12, 13 and so on. So I have 13 blocks free, but if I want a file which will occupy 10 blocks I cannot allocate I have to now go and compact this file system. So this is another, so file cannot grow and we will land up with lot of fragmentation.

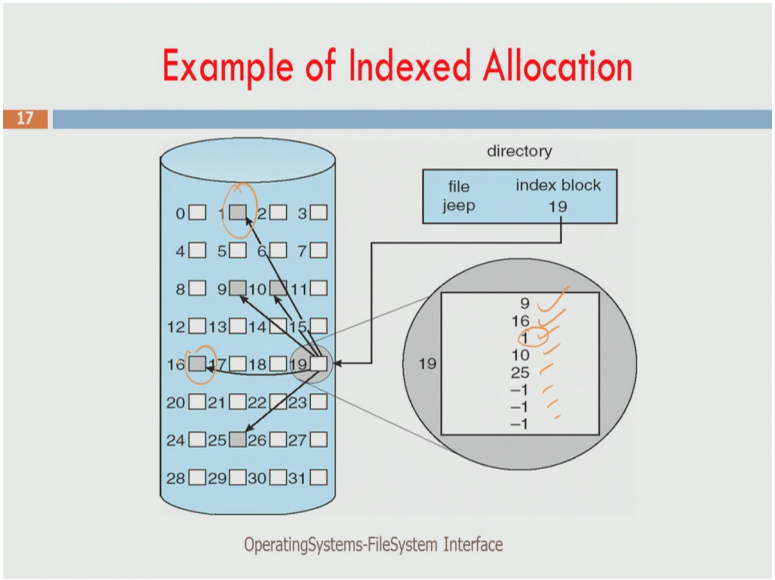
(Refer Slide Time: 5:23)



Now linked to allocation is you start from somewhere, so I start, there is a file called jeep. So this is file called jeep it will start with 9 then there is a pointer there which will take it to 13 from 13 to 16, 16 to 1, 1 to 2, 2 to 18 and then 26 and so on. So it will end up 25 and so on. Now the biggest problem is if I suppose I go and cut this link, right a various kind just go and cut this link, suppose this corrupt this particular block, then the entire file major part of the file is last and suppose we have an encrypted file where this is encoded file or compressed file, where compression lossless then I need the entire file to (())(6:21) the file and some completely is (())(6:22).

So the amount of file, so this is failure rate from virus perspective, this is not going to be a very encouraging thing for us, we may lose more data by having linked allocation. The second thing is access, suppose I want to access the 5 th block I have to go like a link list. So it will take more time for me to access. So from a performance point of view from an access point of view and also from a security point of view linked allocation is not that much desired. So I had pros and cons of, but the file can grow in this case. So I had pros and cons of contiguous allocation, I have pros and cons of linked allocation.

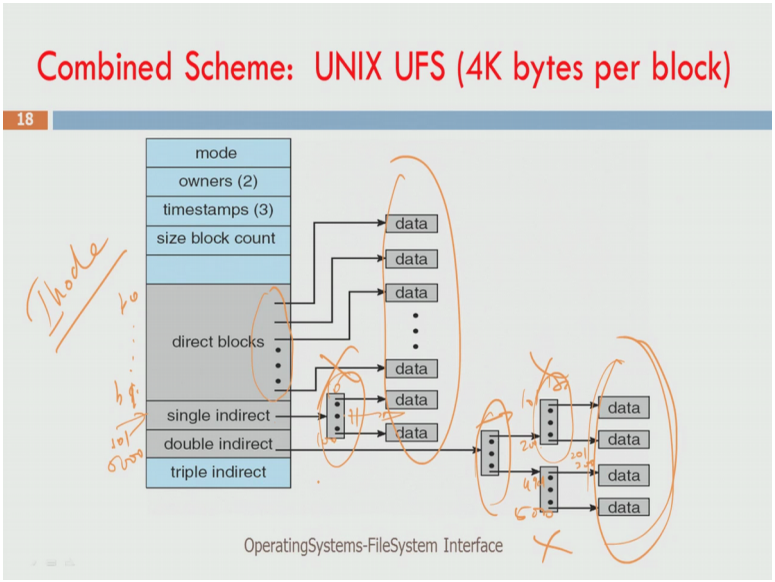
(Refer Slide Time: 7:08)



So let us take the best of both. So I have this what you call as this indexed allocation where given a block I compute a hash function and I go or I just say that the block one is stored at 9, so I maintain an index, so I will say block one is stored at 9, block 2 is stored at 16 so on so fore so, right. So by this if I want to access block 2, I can go to say block 0, block 1,block 2, I can go to the 1 and this is block 2, like this block 3 etcetera. So I can go for everything like this, but then if a file occupy 4 MB and each is half 512 bytes, the number blocks it will

occupy is quite high. So the index itself will become a bigger file for you to manage, okay. So just an indexed allocation the point is, this can grow I could have random search and I could have also growth here, but the thing is the size of the index will become quite large.

(Refer Slide Time: 8:08)



So that is why they have unix has come out with a unique scheme. It is a combine index plus storage allocation. So this a multi-level index. So this is the Inode. So as I told you this an identification node of file, wherein some of the blocks are direct blocks. So block say let us say block zero to say block 9. (())(8:41) sake of this, they are direct blocks. So when I take a file say, the first 9 blocks if I want to access any of the first 9 blocks, I come to this Inode go to this entry and go to that bock and get the data. Then from 10^{th} onwards, I do something called single indirect, so if it is in the 10^{th} block I go here and this will have a point to a block, in this I will store information. So here 0 to 9, in this entry I will store from 10 to say 100 or whatever (())(9:13). So if want to access the 10 th block I come here to this single indirect stage, I go here into this, suppose I want to access 11^{th} block for example, so I come to the single indirect, go to this place and here I will look at the 11^{th} block and that 11^{th} block data will access (())(9:34).

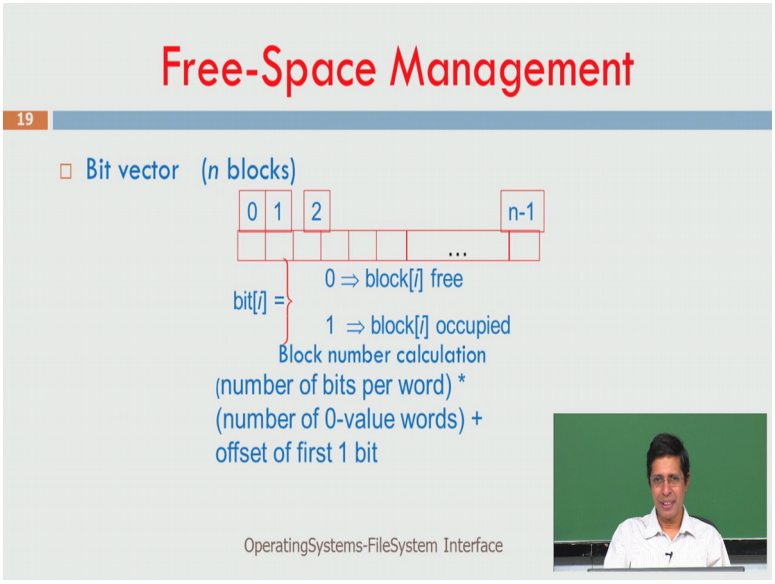
So let us say this will handle till 10 to 100. So the 101 to 5000 I go and do this double indirect. What is double direct? When I say 101 I will go here that will have a set of pointers from there I will go here and so 101 to say each can store 200 like that 201 to300 like that. So some 4901 to 5000 will be stored here. So if I am falling within 101 to 5000 then I go to this block then to another, this will give me pointer to a block from there I will go in that block 101 to 200 will be stored and 201 like that that will have 100 blocks like so.

So this is how I access then there is triple indirection also. So if I am accessing the first few blocks within a file that will be very quick for me, but if I start accessing something beyond that I will go single indirection then to double indirection. So by this if you see in a single Inode of a very given fix at size I could store information about the entire file, which is very huge and that information is which there in the Inode then it is spread across several blocks as you see here this block this block, this block, all of them store information about data how the data is (())(11:06), right. By this and these are all the other things that I am marking now are all the data blocks, right.

So in this structure, we have one parent structure which we called as the Inode or the identification node, which contains details about where the data is stored in this form for the first few blocks, it has a pointer to that data block itself is called direct blocks, then there is a single indirect where it point to a block which has information about the data. Double indirect, it points to a block which again points to a block, which has information about data as you see here, it points to a block which points to another block set of block which has information about data.

Triple indirect it points to a block which has point to a set of blocks which again point to set of blocks which will have information about data, by doing this please note that a large file, the complete detail about a large file would be captured in this scheme. So that is very very important. So this is a combination of indexed or hierarchy of indexed allocation.

(Refer Slide Time: 12:17)



The other important thing is the free space management. How do you manage free space one of the ways by which I could end up with a denial-of-service on your server is to go and go and (())(12:27) with your free space management as long as new process wants to come in, if you do not find free space your system basically hangs. It a basically reject that process, right.

So there are different varieties of free space management, one of this is a bit vector, we says that for every block, a free space management on your disc, so I cannot basically even write a file into a disc, right. So when a process gets into execution and it needs to open a file if it cannot find a file enough free space it cannot basically store, we cannot copy, right. So nothing could be possible, a denial-of-service can basically happen with this free space management. So a disc as say n blocs mark 0 to n minus 1 I have n bits and that n bit if bit, so for every block I have bit, if that block is zero that means it is free and if the it is one it is occupied, okay.

The some of the drawback of this is that if I have lot of blocks then I need lot of bits to maintain that. So there is a need for and extra space, but it is easy to get contiguous file, suppose I want maintain a contiguous file allocation, contiguous space allocation on a discussion. Now suppose I want 10 blocks to store a file in a contiguous way then I should need I should go through this bit vector and find out 10 conjugative zeros, if I find 10 conjugative zeros that means I can allocate that file there.

So to find out contiguous space in the disc is very easy in the case of this bit and I could I could also maintain the link list which will tell you which are all the blocks that are free,

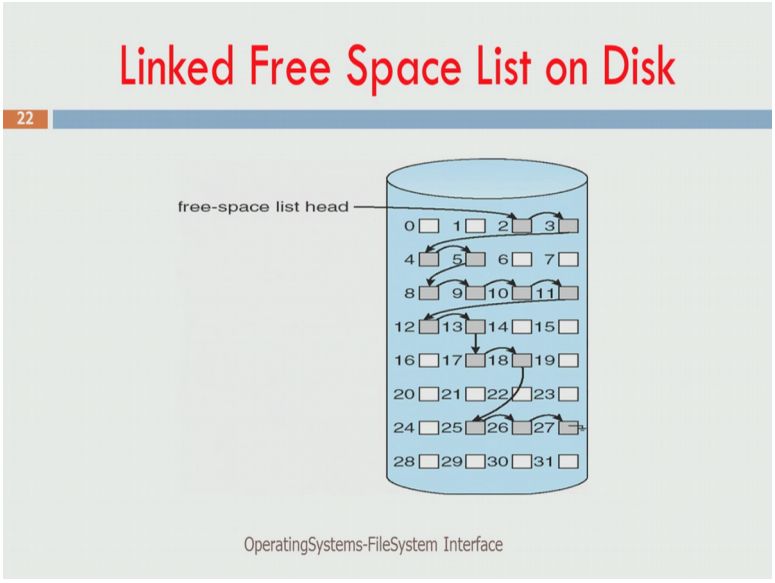
okay. So that is also easy, then I can also group I can count all these things, because it is all bit operation. So I can do lot of things on this.

(Refer Slide Time: 14:15)

So the point is that, this bit map that we have talked off which basically indicates we have what is a free space available. A copy of that bit map will be part of a memory, because whenever I want to create file, I will not go and read from the disc, I would (())(14:32) read it from the main memory. So this bit map or this file free space management structure is a part of your in memory disc. So one of the things the major challenge, the moment I have 2 copies, one at the disc and one in the main memory , there should be a coherency you should not be that, in the disc, it is said it is free, but in the main memory it says, it is occupied or vice versa.

So these type of coherency issues has to be handled and so one of the important aspect of operating systems is basically to handle this type of coherency issues between an in memory structure and on the disc session (())(15:14). So there is reader from the disc the details are copied into the main memory and for performance issues as I repeated (())(15:23) earlier and that coherency should be maintain. So there should not be some bit say two one (())(15:30) here equivalent bit set to 0 there vice versa.

(Refer Slide Time: 15:33)



So normally a free space will also be maintained as link list. So whenever I need file I will I can just go to that free space listed. Now I want so 2, 3, 4 and 5 arte all free in this case. So I need say 2 blocks I will go to the free space I will get that too and then I will make the head now point 2, 3, so **so** quickly I can retrieve space. So this is another thing.

So what we have covered so far in this 12 modules are the basics of operating system which is necessary for you to understand, administration of operating system, in a great detail and I also gave some amount of notions of information security, which will which you need to keep in mind when you start administering this operating system module. So we have done a very quick introduction to these aspects, it is no way exhaustive operating system itself is a one semester course in any of the B. Tech program, we have just spent 12 modules 3 modules per hour 4 hours. It is just (())(16:43) one 10 th of what we get(())(15:55) one 10th or one 15 th of what we get when we teach this operating system as a course.

So I expect that you go and look at information security 2 course also, where you get lot more details about and I have given pointers wherever you have to go and look into information security 2 course and also, some more reading material of standard textbooks which will make you good in operating system, but as far as this course is concerned, for the assignments and also your exam and your certification is concerned whatever I have covered here is sufficient, whatever I have given as pointers is for you to get more in depth knowledge about this whole stuff, right. So that is very very important I am sure none of you are going to do this course just for the sake of certification, because the certification is important, but I am

sure your major interest in doing this course is for gaining more knowledge in depth understanding.

So we have given those pointers, I hope you would follow those pointers and do much more in depth study, even after you complete this assignments and you know finish this exam for this course, but whatever I have told here in terms of extra study and extra look out is very important when you want to become a security engineer for (())(18:10). So **so** that pointer also we have given. So with this my colleagues , Mr. Vasan (())(18:16) and Mr. doctor shankaraman will be covering some of the aspects of administration of the operating system, typically server environment and the network environment, I wish you all the best and see that many of you get certified and I also look forward to lot more interactions on the support portal where you have queries you have very good teaching assistants Dr. subhadra and Mr. prasann kartik (())(18:48) will be also responding to your queries , I want to see lot of query is coming up and lot of discussion happening in this.

This a very very important field and there is a great future about economically and academically and research wise, so people want to do some research program like ph D or MS, information security is a very very important area and of course, if you are looking at the this is a serasvati component, if you have looking at the Lakshmi component of course, there is a lot of job opportunities that will come up very shortly with lot of in a (())(19:33) session happening across the world and importantly in our country, we go into cashless economy, I think information security will play a very crucial role and taking these type of courses very seriously will give you a lot more a secure future for you to have a very good future. So I hope this course will be a good stuffing stone in the direction, thank you very much.