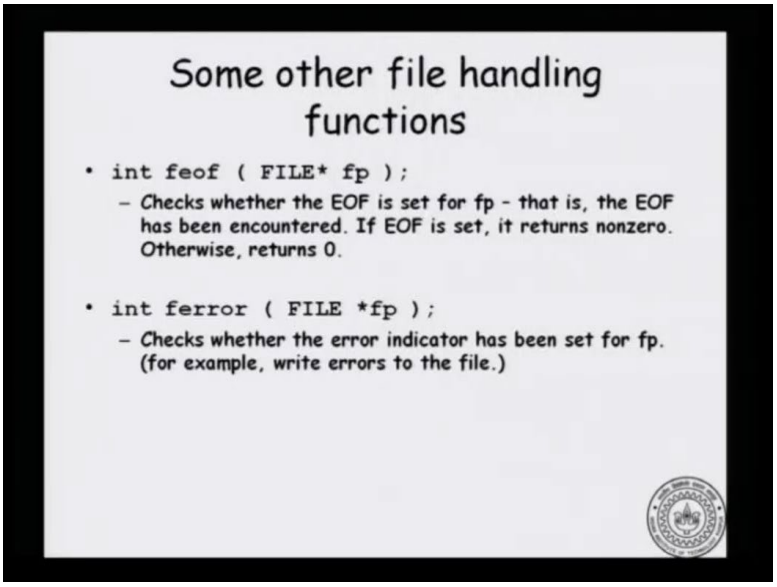
Introduction to Programming in C
Department of Computer Science and Engineering

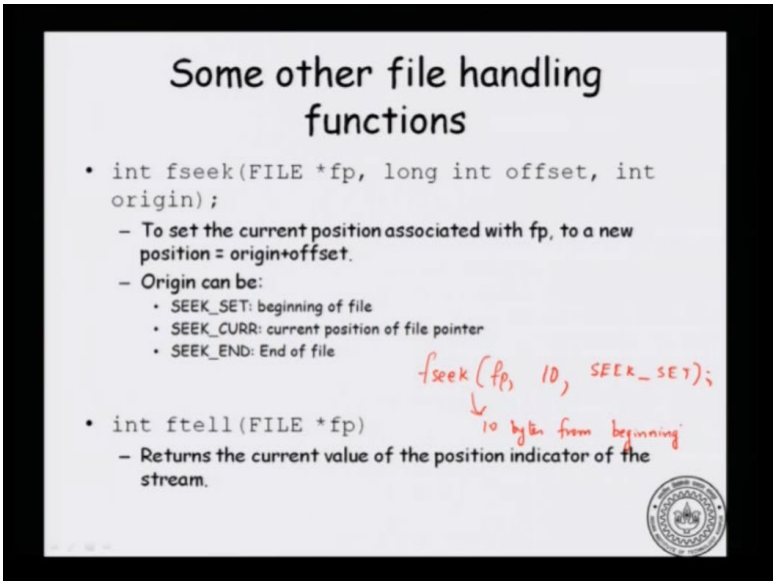
(Refer Slide Time: 00:06)



In this video we will see some more common file operations; these are by no means the only file facilities that C provides you, but in common programming practice these are the functions that people of in use. So, we have seen this in the code that we wrote. The first function is feof, and then it takes a file pointer. What it does is, it is checks whether you have encountered end of file while operating on fp. So, maybe you are trying to read the file, and you have already reach the end of file. So, if you have already reach end of file, that is if EOF is set, then feof returns a non zero value. If feof is not set, that is you have not completed the file yet by seeing end of file, then feof returns 0. So, in order to check whether a file has still has some data, you can just say not of feof fp. So, that will check for the fact that the files still has some data.

Now another useful file function is f error. So, the f error function what it does is it takes the input file pointer, it takes the file pointer *fp, and checks whether you encounter some error while reading the file. So, error may be of many kinds, for example you are trying to write to a read only medium like cd or maybe you trying to write to a file system, and the file system is full. You are trying to write to a hard drive and the hard drive is full. So, then you might encounter an errors. So, there are various errors that you encounter in files operations, and ferror checks for some of these errors. So, if the error indicator has been set for fp, then ferror returns a non zero value, otherwise it is says 0.

(Refer Slide Time: 02:14)

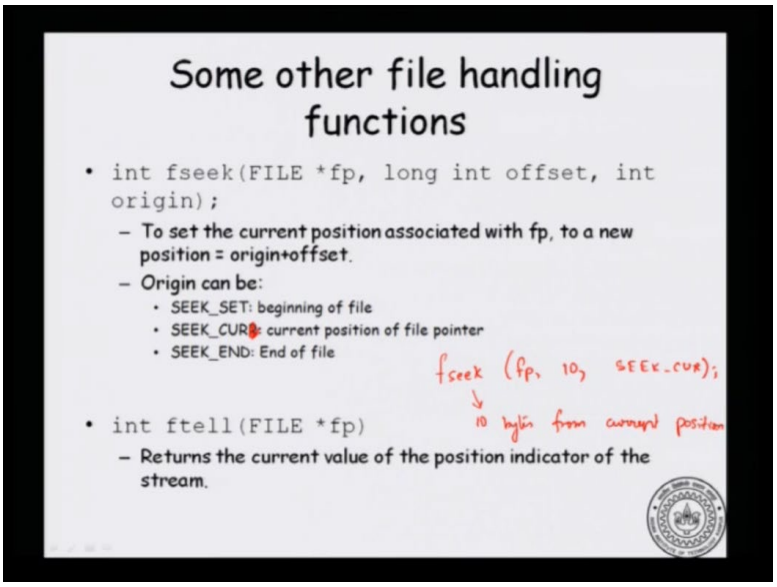


Now here is some couple more interesting functions whose who they are very useful, and a call by commonly use whenever we deal with files. For example, we can have something called fseek. fseek is a function which allows you to start reading from, our start writing to arbitrary locations in the file. So, often we may want to read into the 10000 byte directly, and we do not want to be bother with reading the first 9999 characters discarding them, and then coming to the 10000 th character. This may be lot of wasted time. It will be more convenient, if I can directly jump to the 10000 th location in the file. So, is there a function that allows to you do to that yes, there is just thing call fseek. Now what it takes is the file pointer, and it takes two arguments; one is known as an offset, and the other is known as the origin.

So, let us look at the offset and the origin in greater detail. So, suppose I want to read from the 10 th byte of the file. So, I could say fseek, and suppose by file pointer is fp, I will just say let say I want to read from the 10 th point from the beginning of the file. What I can say is SEEK_SET. So, if I do this what will happen is that? It will start from the beginning, SEEK_SET is the beginning of the file. So, it will add 10 bytes to from the beginning of the file, and it will start from there. So, if I know that I want read from the 10 th byte, then I can say that start from the beginning of the files SEEK_SET says origin of the beginning of the file plus 10 bytes. So, this is 10 bytes from beginning.

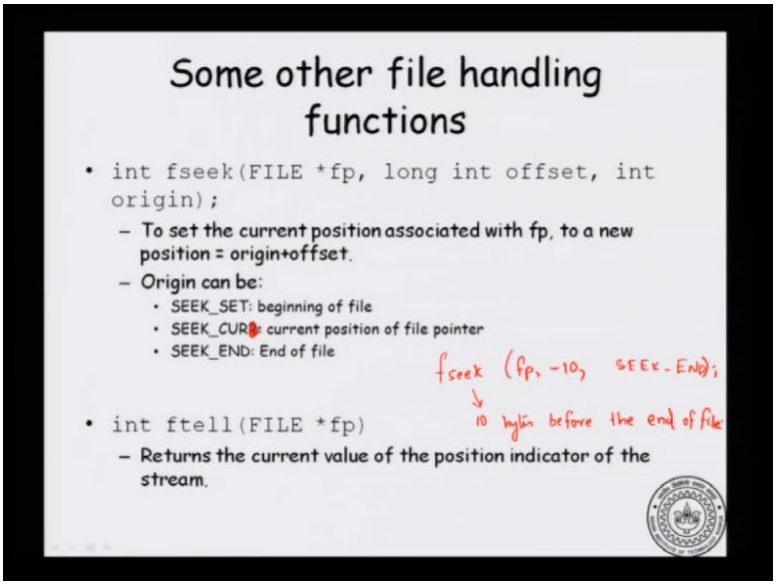
Now, there are other situations, for example you might want to say that I want to start from the 10 th byte from the current location. I have already read many bytes. Now I want to skip the next 10 bytes.

(Refer Slide Time: 05:12)



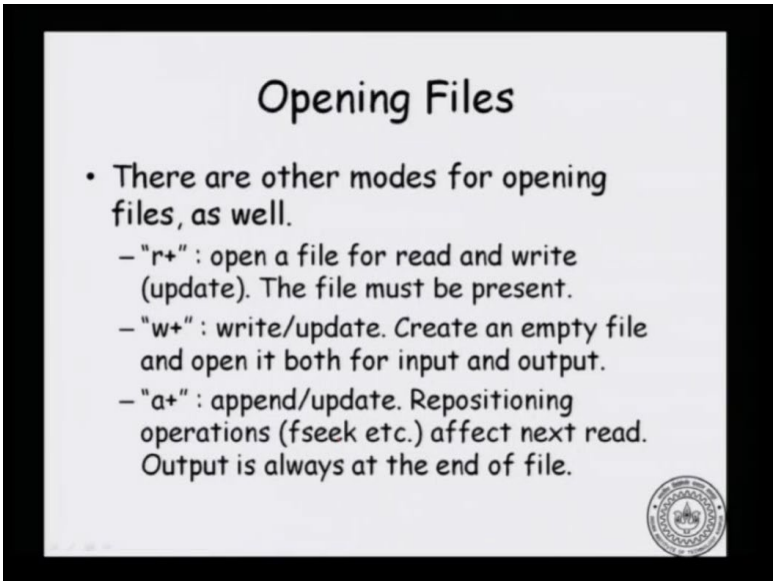
So, is there way to do that again what you can do is, if you say fseek f p and let us say 10 itself, but SEEK_CURRENT. So, there is a typo over here, this is just CUR. So, if I say this, then what I need to do is what it will perform is, it will say 10 bytes from the current position. So, I have already rate 100 bytes from the file, and then I say fseek 10 bytes from the current location. What it will do is jump to 110 th location.

(Refer Slide Time: 06:07)



Now I could also say something like... So, here is a very common situation, I want to start reading from the 10 th byte from the end. So, I want to regardless of the size of the file I want to jump to the end, and then rewind 10 bytes and start from there. So, in that case I can say the origins SEEK_END. So, that is the end of the file and where do I start from SEEK_END plus something does not make any sense, because it is it will refer to something that does not exist in the file. So, you could say SEEK_END -10. So, this is 10 bytes before the end of the file. So you can use fseek in several ways and is a very convenient function, because it allows you to jump to arbitrary location in the file. And it will work as long as the target location origin plus offset is a valid location in the file. Now there is also something called ftell, which will tell you the current position in file. So, if it will take a file pointer as the argument *fp, and it will return you where in the file your currently at.

(Refer Slide Time: 07:35)



So, with this let us take a look at a few more modes in the file operations. So, when you open the file we saw that you could open it in mode r w a. Now there are also some other special modes that see give see, for example there is something called r+. This says you can a open file for reading and writing. So, this is essentially an update mode. w+ will be write an update. So, create an empty file and update that file. And there is something call a + which is appended update, this is somewhat strange. If you do any fseek after you open the file in a + mode, then the read will be effected. So, suppose I am at 100 th location, I have write 99 bytes, I am at the 100 th byte. If I read, if I now do an fseek to 10 bytes ahead. So, now I will be at the 110 th byte.

Now there are two possibilities now, I can read from here or I can do and fprintf, fscanf will start from the 110 th byte, it will be obey the fseek. fprintf will always print at the end of the file. So, that is the append part of it. So, fprintf is always output is always at the end of the file, and reading will be depended on any fseek that you do. So, fseek will never affect the where you print, it will always be the end of the file. So, a + is a very special for it. These are some additional file operations that you might find useful while coding in C.