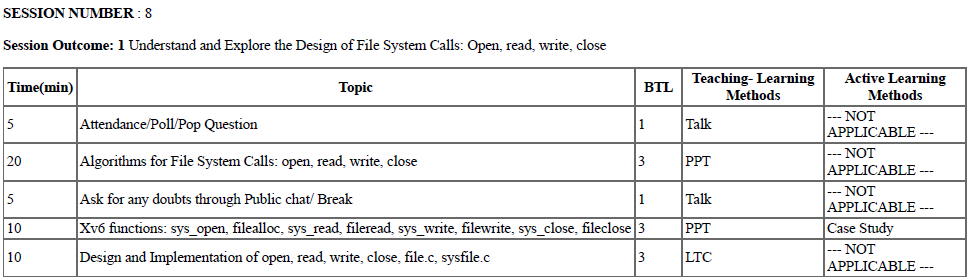
Session-8 Lecture Notes

Topic: **File System Calls**

**Session Plan :** 

**Learning Outcomes:**

* To understand open ( ), read ( ), write ( ) and close ( ) systems calls.
* To explore the above system calls in xv6 as case study.
* To understand the design and implementation of sysfile.c

**Concept of File Descriptor:**

A *file descriptor* is just an integer, private per process, and is used in UNIX systems to access files.

It is dynamically generated by the kernel when you call open( ) system call (or certain other system calls).

When a file is opened, the file descriptor is used to read or write the file assuming you have permission to do so.

The descriptor is identified by a *unique non-negative integer*, such as 0, 12, or 567.

At least one file descriptor exists for every open file on the system.

The first three user file descriptors (0, 1, and 2) are called the standard input, standard output, and standard error file descriptors.

Processes on UNIX systems conventionally use the standard input descriptor to read input data, the standard output descriptor to write output data, and the standard error descriptor to write error data (messages).

***Syntax:***

***fd =open(pathname, flags, modes);***

***Or***

***int fd = open(pathname , flags , modes);***

*Here,* ***pathname*** *is a file name,* ***flags*** *indicate the type of open (such as for reading or writing), and* ***modes*** *give the file permissions if the file is being created.*

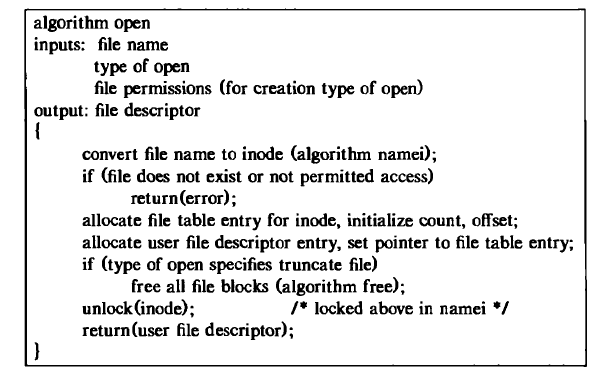
***open ( )* file system Call:**

* The *open system call is the first step a process must take to access the data in a* file.
* The syntax for the *open system call is*

***fd =open(pathname, flags, modes);***

* Here ***pathname*** *is a file name,* ***flags*** *indicate the type of open (such as for reading* or writing), and ***modes*** *give the file permissions if the file is being created.*

***Algorithm:***

******

**read () System Call:**

* The syntax of the read system call is

***number = read(fd, buffer, count)***

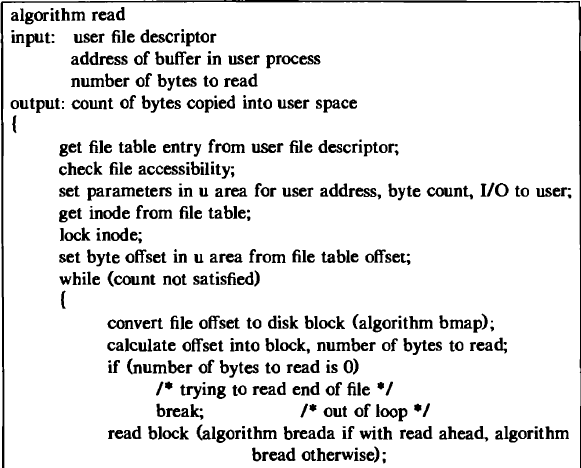
where

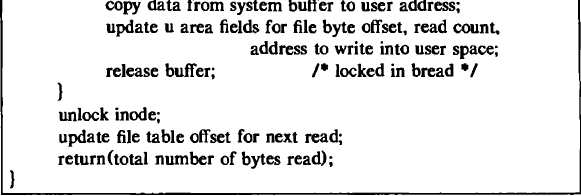
**fd** is the file descriptor returned by open

**buffer** is the address of a data structure in the user process that will contain the read data on successful completion of the call

**count** is the number of bytes the user wants to read, and number is the number of bytes actually read.

**Algorithm:**





***write() System Call :***

Syntax :

***number = write(fd, buffer, count);***

where the meaning of the variables *fd, buffer, count, and number are the same as* they are for the *read system call .*

***close() system Call :***

Syntax:

***close (fd);***

where *fd* is the file descriptor for the open file.

* The kernel does the *close* operationby manipulating the file descriptor and the corresponding file table and mode table entries.

**Xv6 Case Study :**

* *sys\_open*
* *filealloc*
* *sys\_read, fileread*
* *sys\_write, filewrite*
* *sys\_close, fileclose*

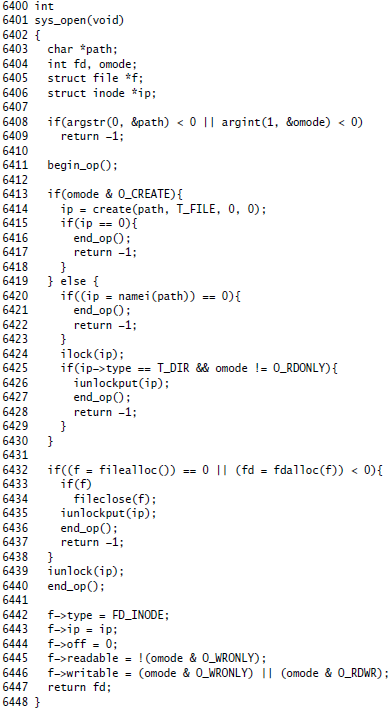
**Design and Implementation of sysfile.c :**

* Its an xv6 file that consists of functions of various file system calls.
* Its available from Sheet No. 60-65 of xv6 code manual.
* It contains the file related functions like *sys\_open*, *filealloc*, *sys\_read*, *fileread*, *sys\_write*, *filewrite*, *sys\_close*, *fileclose*

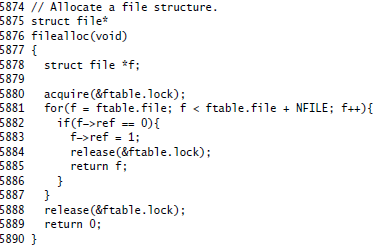
**sys\_open:**

* *sys\_open()* system call opens the specified filename, using the program associated with the corresponding file type of filename.
* The behaviour of this command is the same as that of double clicking on filename in the Windows Explorer.
* For example, if filename is " c:\mydata\sales.xls " and the .xls extension is associated with Microsoft Excel, sys\_open("c:\mydata\sales.xls") will open Excel and load the file.

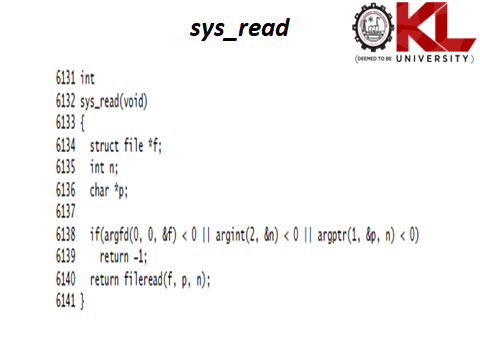
**Xv6 function of sys\_open:**



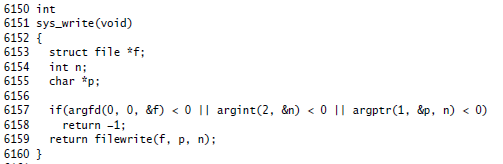
**Xv6 function of *filealloc* :**

****

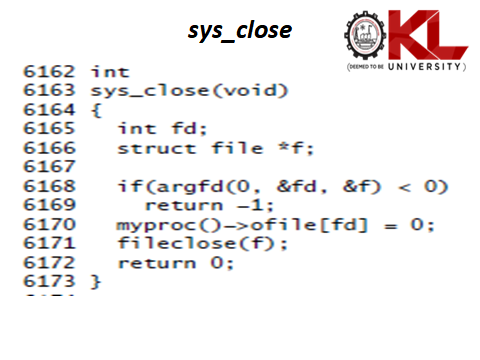
**Xv6 function of sys\_read :**

****

**Xv6 function of sys\_write:**

****

**Xv6 function of sys\_close:**

****