Below the sketch of the Unix facilities for file system calls is shown consisting of: two tables; table entries; and primitive operations that work on these tables.

Tables:

- SysT: System wide file-table
- PerT: Per-process file-table

Fields in a Table-entry corresponding to a file:

- SysT:

  - File name
  - Owner name
  - Type
  - Location in disk
  - Size
  - Time/date
  - File open count

- PerT:

  - Access control
  - Read/write pointer

Some Primitive operations:

- RD(fileName) # Read the directory information for a file from # the disk, and put them in a Table above.
- WD(fileName) # Write back the information of a file from Tables # to the disk.
- SF(Table, fileName) # Search a Table for file name and return the Table entry number for that file (if file exists in Table).
- DT(Table, entryNum) # Delete the Table entry by giving its entry number.

Tasks to do:

**A. [6 Marks] Fill the fields of the Tables with file attributes: file name; owner name; type; location in disk; size; access control; time/date; read/write pointer; file open count**

Provided answer above.

**B. [6 Marks] Explain where is a file descriptor "fd" in the above operations and why the operating systems use file descriptors to handle file operations.**

Primitive operations RD (read data from file) and WD(write data from file) would use file descriptors "fd" to uniquely identify a file opened by a process. When a process opens a file, the os creates a file descriptor that represents that file for that process. The file descriptor is used by the process to access and operate on the file. In this scenario entryNum would be the "fd".

**C. [8 Marks] Use the above primitive operations, and modify Table fields above for a file to perform the following file system calls.**

**i. create-file(fileName) # Note: create-file does not open the file**

1. In the SysT table, in an empty entry, create a new file with the following information by setting the fields:
    - File name = “fileName”
    - Owner name = “ownerName”
    - Type = “extension of the file (ex: .txt or .doc)”
    - Location in disk = “A location on disk allocated to the file data”
    - Size = 0
    - Time/date = current time/date
    - File open count = 0
2. Use the primitive operation “WD(fileName)” to write back the information for FCB to the disk

**ii. open-file(fileName, Read-and-Write)**

1. Using SF(SysT, fileName) we search the table to find the file which identifies as “fileName”, increment the open file counter, and put a pointer to this position into an empty entry found within the table PerT. After this we set the read/write access control along with the read/write pointer into the entry given in table PerT. Lastly, we return the entry number of PerT as the “fd”.

2. If the file doesn’t exist in the table SysT then we use the operation RD(fileName) to read its FCB from the disk into an empty entry in SysT. We then initialize the open file counter to 1 and proceed as before in order to obtain and return the “fd”.

**iii. write-file(fd, Buffer, 100)**

Use “fd” to access the PerT entry and get the SysT entry number to access the FCB of the correct file. Get the pointer of the location of the file on the disk from the FCB. Get the read/write pointer from the PerT table. Group 100 words from the Buffer into the disk block data and write the blocks of data into the disk blocks and then update the read/write pointer.

**iv. close-file(fd)**

Use “fd” to access the correct PerT table entry and get the correct SysT entryNumber to access the FCB of the file. Delete the PerT entry using DT(PerT, fd). Decrease the file open count in the SysT entry and get the name of the file (fileName), if the count becomes zero, then delete the SysT entry as well using DT(SysT, entryNumber). Write the FCB from SysT to the hard drive using WD(fileName).

**v. delete-file(fileName)**

The file is already deleted. Search in the SysT table using SF(SysT, fileName) to get the FCB that corresponds with the file. Delete FCB in the SysT table