Report of Assignment 8

**Description**

The aim is to develop a simple  in-memory file system for Xinu. We are supposed to implement a single level directory with the functionality for open, close, creat, seek, read and write.

**Functionality**

Implement a single level directory with the functionality for open, close, create, seek, read and write. It opens the file only in the create mode .And there are no nested directories.

|  |  |
| --- | --- |
| **File / Function** | **Description** |
|  |  |
| fs\_open | This function receives a file name and the mode in which the file should be opened as an argument. It checks in the directory list if a file of this name exists or not. If there is an entry in the directory list with the same name, then its status is checked if the file is already open or not. If the file is found to be open then a system error is returned. Else if the file is not found to be open then the state of file is changed to OPEN in the mode passed as the argument. Also an entry in the open file table is made for this particular file. The function returns -1 if no file with the mentioned name is found in the table .We also make an entry in the open file table and set it’s various parameters to point to the file. |
| fs\_close | We simply set the status of the file as FS\_closed in the directory entry table for the file. And we free the memory space in the open file table. |
| fs\_create | We take the file name and mode as arguments. We check if the mode(only create mode) is correct or if the file already exists . If either of these conditions are true then an error is returned .Else an index block(inode) is assigned to the file . An entry is made in the directory table regarding the file .Then fs\_open() function is called . |
| fs\_seek | We take the file descriptor and the offset as arguments. After we have accessed the base address of the file we set the read head to the desired position by adding the offset to the base address of the file .It’s also checked if desired position calculated after adding the offset is greater than the length of the file or not . If it is then an error is thrown. |

|  |  |
| --- | --- |
| **File / Function** | **Description** |
|  |  |
| fs\_read | It takes the file pointer and the number of bytes as the argument .It checks if the file is open or the mode passed as argument is read or not. We check if the file pointer position is greater than the number of characters or not. If it is, then we return an error, else we call the function bs\_read to read the characters and return the characters read. |
| fs\_write | It takes the file pointer and the number of bytes as the argument. It checks if the file is open or the mode passed as argument is write or not. If it is then we return an error else we call a function bs\_write to write the desired characters to the file. The size of the file is updated by the number of characters having been written in the open file table. |
| fs\_mount | It resolves the internal name space. Mount function allows us to add mappings to the prefix table. It takes in three arguments: a prefix string, a replacement string, and a device ID. If any of the arguments are invalid or the table is full, mount returns SYSERR. Otherwise, it increments nnames to allocate a new entry in the table and fills in the values. |

**Tasks and responsibilities**

|  |  |
| --- | --- |
| **File Name/task** | **User ID** |
|  |  |
| fs\_open | Shalabh |
| fs\_close | Hemant |
| fs\_read | Shalabh |
| fs\_write | Hemant |
| fs\_create | Shalabh |
| fs\_seek | Hemant |
| fs\_mount | Shalabh |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |