Bivas Maiti
bmaiti@iu.edu

Assignment 6: File Systems

In this assignment, a file system is implemented in the XINU Operating System. XINU does not originally have any file system support. This file system is designed keeping the ext2 file system of linux as a reference. The basic functions of a file system like read, write, create, open, seek and close are implemented. All the files exist in the root directory in the file system. The file system exists entirely on the RAM. The functions implemented are described below:

Fs_open

Arguments: filename, flags. Returns: file descriptor

This function takes the name of the file and returns the file descriptor for that file. Internally, it searches in the root direcrtory for the file. If it is present, it simply returns the file descriptor of that file. But if it does not find the file, it creates a new file of size 0 bytes, opens it and returns the file descriptor of that file.

Fs_close

Arguments: file descriptor Returns: status of execution

This function iterates through the file descriptor table, oft, and tries to find the file which needs to be closed. If it finds the file, it updates the state of the file as closed in the table, and returns OK. If it doesn't find the file, it returns SYSERR.

Fs_create

Arguments: filename, mode Returns file descriptor

This function is for creation of new files. First it creates a new inode, with attributes of the file, and file size=0. Then it creates a directory entry(dirent) object for the file. Then it inserts these values in the file descriptor table, and returns the file descriptor index value. It also updates the next open FD in the FD table.

Fs_seek

Arguments: File Descriptor, offset Returns : status of execution

First it checks if the offset is larger than the size, if it is, SYSERR is returned. If not, it sets the offset by the provided value in the fileptr attribute of the FD struct.

Fs_read

Bivas Maiti
bmaiti@iu.edu

Arguments: File Descriptor, buffer, nbytes Returns: number of bytes read or SYSERR.

This function first reads nbytes number of bytes from the file with the FD provided. The bytes are read from the file from the offset of the fileptr. So if the file is of 1024 bytes, and the fileptr value of the FD is 100, it will read nbytes after 100. After reading nbytes, the function also increments the value of the fileptr with nbytes. If everything happens successfully, size of the file is returned.

Fs_write

Arguments: File Descriptor, buffer, nbytes Returns: number of bytes written or SYSERR.

This function is used to write into a file. First it checks if the file is already present in the FD table. If not, it returns SYSERR. Now it checks the size of the file. If the size is more than 0, then there are several datablocks associated with the file. The function first clears those datablocks. After that it finds free datablocks and writes into them. It also changes the size attribute of the file and the fileptr attribute of the FD entry.

Lessons Learnt

1. Concept and implementation of File Descriptor- File descriptor acts kind of like a local cache of all the open files and devices and allows quick access to the open files.
2. Implementation of Data Blocks and Inodes: In this assignment I learnt how inodes are created and maintained as a part of a file system. Inodes have array of blocks which then refer to the actual data blocks.
3. This assignment can be further extended to include features like directory support, second level of indirection from inodes in case of larger files.
4. Implementation of Bitmask array: In this assignment I learnt how the bitmask array is maintained to check if a particular data block is used or empty.