CS3500 - Operating System, August 2022

Lab 8: File System

Due Date: Friday, 2th November 11.59PM in Moodle.
Demo Date (if required): 4/11/2022

The objective of this assignment is to implement a simulated file system. Your simulated file system will be fully contained in a single file, stored in your working directory, and will be called "Diskfile". The Diskfile will contain directory disk information, free block information, file data blocks, etc. The name of the diskfile and the size of the disk file (in KB) will be input as command line parameters. Your program will accept system commands from the user as described in the sample session.

Input values: Diskfile, Disk size (D in KB) & one or more files. The size of each disk block is fixed to be 128 bytes. Thus, D = 10, indicates a 10 KB virtual disk with block size of 128 bytes, and there are 80 blocks in the disk. The disk block address size is 32 bits.

Diskfile information: The program needs to maintain data of,
1. The disk will contain a set of blocks dedicated to maintain disk directory information. The number of blocks reserved for this information is left to the implementer.
2. The free space data, free space maintenance data structure to be used is also left to implementer.
3. The file data blocks

The disk will contain a list of files and directories, with a maximum of two levels in the directory tree. The root of the disk is denoted as '/'.
For example, the disk can have contents: /f1, /f2, /d1/f3, /d2/f4
Here, the above entries denote two files f1 and f2, and two directories /d1 and /d2, containing one file each.

File Information: A file is viewed as a sequence of bytes, with the first byte numbered 0. File is allocated space on the disk, one block at a time. All references to files in this assignment will be by their fully expanded names (i.e. including complete directory tree information), as in /dir1/file2.h. For filenames, extensions are optional and can include up to three characters.

The **inode** structure for each file contains the following information:

1. Directory name or Filename: Up to 16 bytes total for a fully expanded filename (including the two '/' characters). The directory name can have up to 4 characters (bytes).
   Thus, di4, dir3 are valid directory names, while direc3 is not; Likewise, /dir3/filename.c is valid since it contains 16 total characters.
2. Type: Directory (value = 1) or File (value = 0): Use 1 byte to store this value.
3. File size: recorded using 4 bytes. Theoretical maximum file size is 232 bytes, but in practice, the file sizes for this project will not exceed 4,480 Bytes.
4. Date Created: Output of date command of Linux - requires a 28-byte string.
5. Date Last Modified: Output of date command of Linux - requires a 28-byte string.
6. Direct block addresses: 3 4-byte integers, that will point to the first three data blocks of the file.
7. Index block address: 1 4-byte integer, that will point to one index block, which can contain up to 32 block addresses (Since each block size is 128 bytes and each block address needs 4 bytes). These addresses in the index block point to blocks 4, 5, 6, etc. of the file.

Thus, the maximum possible file size = (3 + 32) = 35 × 128 = 4, 480Bytes.

Note that the inode information for each file is stored on the disk in the disk directory information part of the disk. Assume that only one inode is stored per disk block.

Problem statement
You must implement a disk directory structure, which will store information about all files on the directory and the inodes for the files. The specific inode structure and data maintained for directory level information is left open to the implementer.

System commands: On startup, the system waits for commands from user. That is, it is an interactive program that executes the user's commands, which will be from the following list:

- load filename
  Load the file data in the disk, create a inode for file & allocate the blocks.
- delete filename
  remove the file data from the disk, delete the inode for the file & allocate the blocks to free block.
- print filename
  prints the content of the file.
- append filename
  append the content at the end of file. If required allocate the new blocks from free blocks & update the inode.
- display
  prints the name of the files in the disk & its size.
- freespace
  prints the number of available free block & total free size.
- printinode filename
  Print the inode contents for the specified file.
  For example, assuming a block size of 128 bytes and file size of 600 bytes (5 blocks):
  ****
  Filename: /file1.c
  Size: bytes
  Date Created: Thu Apr 13 17:59:55 EDT 2006
  Date Last Modified: Thu Apr 20 16:45:10 EDT 2006
  Direct block values: 23 51 67
  Index block is stored in: 88
  Index block contents: 34 12
  ****

- exit
  Exit the system.

Note: For all relevant commands, if there is no space available on the virtual disk, print an error message and proceed to the next command.