DB101 - DATA MANAGEMENT
Fall 2012

Data Structures Project

GENERAL PROJECT INFORMATION

Course Name	DB-101 Data Management
Project Name	Virtual File System (VFS)
Project Description	VFS is logical view of the filesystem at a higher level of abstraction. The project supports basic operation for file management.
Project Type Version	Individual 1.0
Document last updated at	22-Aug-12

PROJECT DESCRIPTION

A **virtual file system (VFS)** is an abstraction layer on top of a concrete file system that provides the ability manage information about files and folders present in the hard disk. The purpose of a VFS is to allow client applications to access different types of concrete file systems in a uniform way. A VFS can, for example, be used to access local and network storage devices transparently without the client application noticing the difference.

(Refer: http://en.wikipedia.org/wiki/Virtual_file_system)
All operations, file system operations are Unix type.

Glossary of Terms

- File System – A system that provides a unified mechanism for managing data that is organized in the form of files
- File Blocks – Fundamental storage unit into which files are actually stored. A single file may span across multiple file blocks
- File Descriptors – A data structure that contains meta-data information about the file (name, type, location, etc.)

Overview:

File System is basically a large file which stores the data contained in all the existing files. Each file/folder in a file system has a corresponding File descriptor which stores all the information of the file /folder e.g. Size, name, path etc. File system is divided into data blocks of fixed size which store the actual file contents. The files when saved in the file system can be scattered over multiple blocks depending upon the size of the file.

Example:
Suppose there is a file system of 10 Kb divided into 10 data blocks(0-9) of 1Kb each. Suppose there are three files a.txt, b.txt, c.txt each of size 1.25 Kb.
a.txt resides in blocks 0,1
b.txt resides in blocks 2,5
c.txt resides in block 3,4

The file system in this project consists of three parts as shown in the figure.

- Meta Header [Fixed size]
  - Number of file descriptors present
- Header [Fixed size]
  - File descriptors of each file/folder in the file system. These file descriptors would be needed to populate n-array tree for the directory structure and hash table for faster searching for files.
  - There would be a separate free_block_list which contains the list of all the free blocks in the file system.

- File Blocks – Each identified with an index number or an offset.

<table>
<tr><td colspan="5">Meta header (fixed size)
(name of file system, number of used file descriptors)</td></tr>
<tr><td colspan="5">Header (max of N file descriptors + Free list)
(fd_01, fd_02, … fd_N)</td></tr>
<tr><td>Block 0</td><td>Block 1</td><td>Block 2</td><td>.....</td><td>Block n-1</td></tr>
</table>

Key Data Structures used

- N-ary tree
- Hash table
- Linked list
- Binary Search Tree
- Appropriate indexing structure for text-based search

Purpose of Data Structures

- N-ary tree would be used to represent the directory structure of the entire file system. Each node in the tree would be the file descriptor corresponding to the file/folder.

- Hash table would help in searching functionality for files on the basis of the name of the file. Each bucket in the hash would hold the files according to the starting alphabet of the file. Example:

Key values	Buckets
a	aa.txt-->ab.ppt-->ac.doc
b	ba.txt-->bb.doc-->bc.ppt
c	ca.doc-->cb.txt-->cc.ppt
.	
.	
.	
z	za.txt-->zb.ppt-->zc.txt

- Linked list keeps a list of the free data block into which new files can be written.

- BST is used to perform search functionality on files on the basis of absolute path of the file.

- You need to explore and come up with suitable data structure for content-based search too

High Level VFS Operators

1. File System Operators
   a) **create_vfs (vfs_label,size)** – To create a Virtual File System. <vfs_label> is the name of VFS.
   b) **mount_vfs (vfs_label)** – To Mount the VFS path on current directory. This involves loading the file/directory descriptors into various data structures
   c) **unmount_vfs (vfs_label)** - To unmount the VFS. It involves saving index information and tree information back into the hard disk.

2. Directory Operators
   d) **makedir (parent_path, dir_name)** – To create a new directory whose name should be <dir_name> in path specified by <parent_path> where '/' is considered as 'root' directory.
   e) **deletedir (name)** – To remove a file or a directory as indicated by < name>
   f) **movedir (source_dir_name,dest_dir_name)** – To move a sub-tree <source_dir_name> to <dest_dir_name> directory.
   g) **listdir (dir_name,flag)** – List all the contents according to <flag> of the current directory i.e. specified by <dir_name>.

3. File Operators
   a) **create(dest_dir_name, file_name, data)** - Create a new file named <file_name> with <data> as content. in path specified by <dest_dir_name>.
   b) **listfile(file_path)** - View the contents of a file specified by <file_path>.
   c) **update(file_path, data)** – Update/Edit the contents of a file specified by <file_path> with <data> according to <flag>.
   d) **rm(file_path)** – Remove a file specified by <file_path>.
   e) **mv(source_file_path,dest_file_path)** – Rename <source_file_path> to <dest_file_path> file.
   f) **copy (source_file/dir_path,dest_file/dir_path)** – To copy source directory/file to destination specified.
   g) **Export (file_path)** – Export the file from the file system into the hard disk

EVALUATION PROCEDURE

Your software will be evaluated using interaction scripts. Interaction scripts contain a series of VFS operators. A sample interaction script is shown below:

```
create_vfs demo01
mount_vfs demo01
makedir ROOT home
makedir ROOT/home demo
create ROOT/home/demo hello.txt
listfile ROOT/home/demo hello.txt
unmount_vfs demo01
```