Project 5 - Filesystems

4/25/2012

Overview

- Filesystems Background
- GeekOS Filesystem (GOSFS)
  - Background
  - Format
  - Mount
  - Other operations

What is a Filesystem?

Resource Manager

- a means to organize data

What is a Filesystem?

Resource Manager

- a means to organize data (and accesses to that data)

What is a Filesystem?

Resource Manager

- a means to organize data (and accesses to that data)
  - usually, data organized into "files"
    - File - named sequence of bytes (with metadata)
  - usually, use directories to organize files

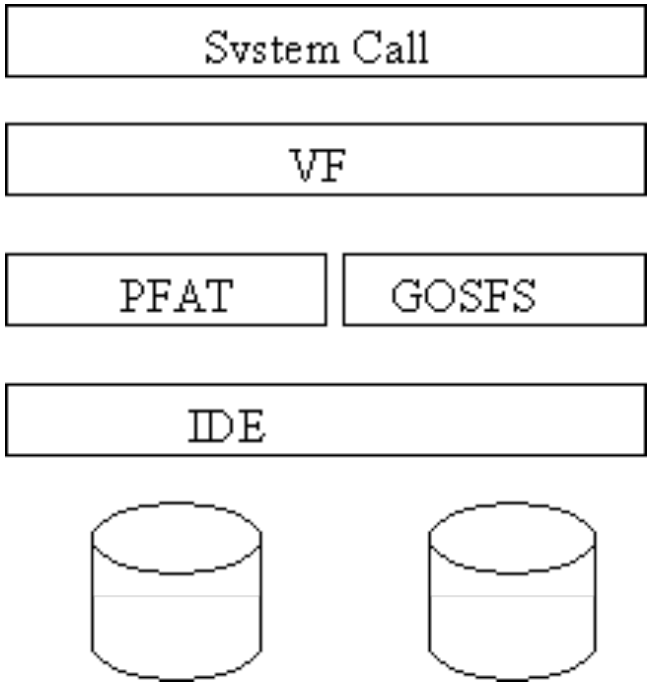
So, in the end, basically just something that manages files, usually in a directory structure.

How to manage files?

- Manage Filesystem
  - Format
    - structure "something" as needed for filesystem
  - Mount
    - "Take the file system from this CD-ROM and make it appear under such-and-such directory" - Wikipedia
- Manage Files
  - Open, Close
  - Read, Write
  - Stat
  - Seek
  - etc.
- How to do this? Answer

Common Filesystem Operations

- Because many ways to organize data, good to have standard ways to use filesystems via abstractions



Summary

Filesystem is something that manages files and accesses to files.

This is done by implementing some basic functions. (From which can build on to do more)

GeekOS Filesystem Background

GeekOS Filesystem (GOSFS)

- On startup detects devices (e.g ide0, ide1)
  - diskc.img (pfat), *diskd.img (gosfs here)*
  - Associates Block_Device with them
- Automatically mounts ide0 (pfat)
- Shell starts
- You will format ide1 into GOSFS format
  - look at p5test.c:ttestFormat()
- Mount ide1
  - use Mount user program provided
    - can also look at p5test.c:ttestMount()
- Use your filesystem to do stuff
  - look at p5test.c

Interacting with Disk

- Block_Read(struct Block_Device *dev, int blockNum, void *buf)

- Block_Write(struct Block_Device *dev, int blockNum, void *buf)

- Writes a SECTOR_SIZE at a time so need to do 8x to write a 4KB block
- This Block_Device is available in all of the functions
  - mountPoint->dev
  - file->mountPoint->dev

Useful data structure

```
typedef struct {
    char name[64];              /* name of file */
    int size;                   /* size of the file */
    unsigned int isUsed:1;      /* is entry active */
    unsigned int isDirectory:1; /* is this file a directory */
    unsigned int isSetUid:1;    /* is this file setuid */
    int blocks[10];             /* 8 are for direct blocks, 1 for indirect and 1 for double indirect */
    struct VFS_ACL_Entry acls[VFS_MAX_ACL_ENTRIES];
} GOSFSfileNode;

/* should fit in one block */
#define MAX_FILES_PER_DIR  (4096/(sizeof(GOSFSfileNode)))

typedef struct {
    GOSFSfileNode files[MAX_FILES_PER_DIR];     /* all of the files */
} GOSFSdirectory;
```

- Directory: blocks[0] contains block number of 4KB block
  - That 4KB block contains an array of more GOSFSfileNodes
- File: blocks[0] to blocks[7] "point" to data
  - blocks[8] "points" to block of more pointers to data (indirect block)
  - blocks[9] - double indirect block

Similar to "inode" if you want to read about it

GOSFS_Format

- Make the disk look like:

<table>
<tr><td colspan="5">SUPERBLOCK</td><td>REST OF THE DISK</td></tr>
<tr><td colspan="5">4KB</td><td>32 MB-4KB</td></tr>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>1024 bytes</td><td>4KB-1036 bytes= 3060 bytes</td><td>32 MB-4KB</td></tr>
<tr><td>Magic</td><td>Root Dir Pointer</td><td>Size</td><td>Free Blocks Bitmap</td><td>Free</td><td>Data
(Not allocated blocks or blocks allocated to directories/files)</td></tr>
</table>

GOSFS_Mount

- Verify superblock
  - Verify magic number
- mountPoint->op
  - need to give function pointers so tha vfs knows how to handle operations (can look at pfat and vfs to understand what is expected here)
- mountPoint->fsData
  - up to you

GOSFS_*

- Modify passed File * as needed
- Read / write contents of disk as needed

Other comments

- VFS does much of the "File" stuff for you, make sure to read over vfs.c to know how your code will fit in
- Project 6 builds on Project 5, will definitely need at least Format / Mount / Open working
- Use p5test.c to test, since doing disk operations, expect server testing to be somewhat slow...
- gosfs.c starts at ~100 lines, not unusual for final implementation here to be 1000+ lines...